APPENDIX C

SOFTWARE DESIGN DESCRIPTION FOR THE HISPANIC NAME SEARCH ALGORITHM (HNA - E)

TABLE OF CONTENTS

SOFTWARE DESIGN DESCRIPTION FOR THE HISPANIC NAME SEARCH ALGORITHM (HNA - E)

1. INTRODUCTION

1.1. Purpose

The current VLDB consists of about 5 million refusal records. The outlook envisions significant growth of the database in the near future and continued growth as more and more data are shared with other Government agencies. Currently, between 45% and 50% of the records have a country of birth from the Hispanic world, about 2.5 million records. These proportions are unlikely to change as the database expands.

Additionally, the character of Hispanic personal names is such that they are both *dense* and *complex*. *Dense* means that there are a relatively few individual surnames that account for the vast majority of surname occurrences. That is, the 500 most frequently occurring distinct surnames account for over 70% of all distinct surnames in the database. The surnames of well over 50% of the records contain only high frequency surnames. Another 25-30% contain at least one of the high frequency surnames. *Complex* means that Hispanic surnames generally contain more than 1 surname, the first of which is the family name, the second a matronymic (FLORES GOMEZ). Approximately 75% of the surnames from the Hispanic partition contain 2 surname stems (not including affixes like DE, DE LOS). Another 23% have only 1 surname stem. (The remaining records have 3-6 stems.)

The frequency of the names, the high portion of the VLDB and the syntactic variation that can occur in these names (inversion of the names, deletion of a name) argue for special handling of the Hispanic name search process.

The most important aspect of this specialized Hispanic name search algorithm is an efficient High Frequency Name Processor. Retrieval of fewer records for evaluation, yet ones that reflect some variation, is the goal of the High Frequency Processor.

The High Frequency Processor (HFP) of the HNA-E system targets the efficient processing of the most frequently occurring records in the Hispanic portion of the database. Early attempts at developing a processor that would handle high frequency Hispanic names had several major weaknesses.

- The earlier processors did not adequately address the characteristics of Hispanic names. In the name of performance, they did not allow for any variation in high frequency names.
- There was only one access method to the high frequency processor, which eliminated the processing of names similar to the high frequency names by the High Frequency Processor.
- Strict, often unmotivated, limitations were placed on the high frequency retrieval process. Little to no spelling or syntactic variation was permitted.
- The number of records retrieved was often extremely high, which resulted in a significant amount of post-processing.

All of these issues have been addressed in the HNA-E design. The HFP will be primarily list-based but the lists are empirically developed. It will identify and store relevant information about names, variants and their degree of proximity and will apply record similarity criteria *before* retrieval.

Low frequency Hispanic names, on the other hand, carry more information value because they are less usual. However, even low frequency names occur with sufficient frequency to challenge the system; the Hispanic database in general is very large. Preprocessing low frequency names will, therefore, also help reduce the number of records retrieved by limiting the search criteria.

1.2. **Scope**

The HNA-E system is intended to provide special and unique handling for names identified as Hispanic by the Automatic Name Classifier (ANC-E). It addresses the problem of highly frequent names to maximize retrieval potential and minimize the impact on performance and handles less frequently occurring names differently to accommodate the greater information content in these names. It also allows for broad variation in low frequency names and identifies potentially relevant records before database retrieval.

The input into the HNA-E system will be the output of the Advanced Name Classifier (ANC-E). ANC-E will determine if a name is Hispanic and therefore will undergo special processing by the Hispanic Name Algorithm (HNA-E). The design description of the ANC-E is contained as Attachment A in LAS Linguistic Memorandum CT970044 (May 30, 1997).

It became clear during the research for this design that the data stores that would be seminal to this system were very large. The Low Frequency Surname Type Data Store, for example, has over 90,000 records in it. Well over 37,000 of these names occur one time in the database; many of these are obvious misspellings or truncations of names. That is, the character strings do not occur in Spanish: *RR*ODRIGUEZ, for example. It is suggested that a program of data stewardship be initiated to increase the efficiency of the system and reduce the storage needed for deviant material. One method of introducing data stewardship at this juncture would be to introduce Base

Records for the database records with errors and make the current database record of the new Base Record.

1.3. Definitions and Acronyms

ANC-E	Advanced Name Classifier for CLASS-E
DELETE	The name segment is completely disregarded in the remainder of the name search process and contributes minimal information to the record evaluation process; do not remove the segment from the record
DISREGARD	The name segment is disregarded in the remainder of the name search process but contributes to the evaluation of the name in the record evaluation process; do not remove the segment from the record
DI_KEY	Digraph Key (low frequency name types)
DI_VAL	Digraph Value (two-place decimal indicating digraph relation of two comparands.
F	Female
FNU	First Name Unknown
FPD	Frequency Path Director
FTI	Frequency Type Identifier
GN	Given Name
GNDR	Name Gender
GNTHR	Given Name Threshold (filter qualification)
GN_INIT	Given Name Initial Key
GN_VAL	Final Given Name Value
HCD	Hispanic Character Data Store
HDM	Hispanic Decision Matrix
HFGN_KEY	High Frequency Given Name Key (SET_ID of the GN_TYPE)
HFGV	High Frequency Given Name Variant Data Store
HFGN_VAR	High Frequency Given Name Variant Key
HF	High Frequency
HFP	High Frequency Processor
HFS	Hispanic Filter and Sorter
HFSN_KEY	High Frequency Surname Key (SET_ID of the HFSN_TYPE)
HFSN_VAR	High Frequency Surname Variant Key (ID_NO of the HFSN_VAR)
HFST	High Frequency Surname Type Data Store
HFSV	High Frequency Surname Variant Data Store
HGI	Hispanic Gender Identifier
HGT	Hispanic Given Name Type Data Store
HNA-E	Hispanic Name Search Algorithm for CLASS-E
HNF	Hispanic Name Formatter
HNP	Hispanic Name Preprocessor
HNT	Hispanic Given Name Type Data Store
HPD	Hispanic Parameter Data Store
HR	Hispanic Regularization Rule Base
HRE	Hispanic Rule Engine
HSE	Hispanic Search Engine
HSP	Hispanic Segment Positioner
HSS	Hispanic Surname Segmenter
HTD	Hispanic TAQ Data Store
HTP	Hispanic TAQ Processor
ID_NO	Identification Number for Segments in Data Stores
INITGN	Given Name Initial Parameter Value

INITNM	No Match Initial Parameter Value
INITSN	Surname Initial Parameter Value
LFDIKEY	Low Frequency Digraph Key in LFST
LFGT	Low Frequency Given Name Type Data Store
LFP	Low Frequency Processor
LFST	Low Frequency Surname Type Data Store
LF_DI THRESHOLD	Low Frequency Digraph Threshold
LNU	Last Name Unknown
LTF	Linguistic Trace Facility
M	Male
NLD	Name Length Determiner
REMOVE	A segment that is conjoined to the name stem is removed from the stem; it will then be marked for additional handling, DELETE or DISREGARD.
RGNDR	Record Gender
RL#	Refusal Code Level Category Number
SET_ID	Identification Number for Related Set of Name Variants
SEGMENT	Name element surrounded by white space
SN	Surname
SNTHR	Surname Threshold (filter qualification)
SN_INIT	Surname Initial Key
SN_VAL	Final Surname Value
SPI	Segment Position Identifier
TAQ	Title/Affix/Qualifier
TAQDIS#1	TAQ DISREGARD Comparand #1
TAQDIS#2	TAQ DISREGARD Comparand #2
U	Unknown/Ambiguous Name Gender
YOB	Year-of-Birth
YOB#	Year-of-Birth Range Category Number
YR	Year-of-Birth Range Data Store

2. PROCESS FLOW

A Hispanic name is pre-processed and prepared for key generation. Prefixes are removed, certain name segments are moved, record gender is determined and other name characteristics are collected.

The processor to which a name is submitted is dependent on the frequency of the surname, high frequency or low frequency. There are multiple entries into the High Frequency Processor, which means that low frequency names that are related to high frequency names can also be treated as high frequency names.

The underlying principle behind the handling of high frequency names is that they retrieve a specified set of variants, all of which have pre-determined digraph values associated with them. This places the processing burden on adding records to the system and reduces the burden at the time of the query. Record retrieval criteria have been defined according to the values of the names and their relative positions in the query string; a query with high frequency names will, therefore, retrieve a smaller set of relevant names. The goal is to retrieve an adequate range of names as rapidly as possible.

Variants of low frequency names will be identified before retrieval based on matching digraph keys. The system will then retrieve exact matches on the set of low frequency names that pass a low frequency threshold.

3. MODULE DECOMPOSITION

3.1. HISPANIC NAME SEARCH ALGORITHM FOR CLASS-E MODULE DECOMPOSITION

3.1.1. Identification

This program is known as the Hispanic Name Search Algorithm for CLASS-E (HNA-E)

3.1.2. Type

This program is a subprogram of the CLASS-E system and will process Hispanic names for both queries and record adds.

3.1.3. Purpose

HNA-E will process input names identified as Hispanic by the ANC-E using techniques that are appropriate for Hispanic names. No names with Last Name Unknown (LNU) will be processed by HNA-E.

3.1.4. Function

The Hispanic Name Search Algorithm for CLASS-E (HNA-E) consists of three program modules:

- the Hispanic Name Preprocessor (HNP),
- the Hispanic Search Engine (HSE), and
- the Hispanic Filter and Sorter (HFS).

3.1.4.1. The HNP will manipulate an input name to generate search keys, generate additional query forms or alias record adds, calculate record gender, collect information about the input name and its name segments and determine the frequency path to which a name will be submitted for processing.

3.1.4.1.1. The HNP will pass an input name to one of two processing paths:

- the High Frequency Name Processor (HFP) or
- the Low Frequency Name Processor (LFP).

3.1.4.1.2. The HNP will generate a set of record criteria and search keys for retrieval of records from the database.

3.1.4.2. The HSE will build the retrieval keys, extract record information relevant to the retrieval and retrieve database records according to the keys and criteria identified.

3.1.4.3. The HFS will evaluate the database records and will prepare an ordered set of records for return to the user.

3.1.4.3.1. The HFS will qualify records based on filtering criteria and parameters.

3.1.4.3.2. The HFS will sort the qualifying database records into an ordered list with the names most closely proximate to the query name at the top.

3.1.5. **Subordinates**

HNA-E consists of 3 major programming modules: (See Pages 7-10 for graphic representations of the processing flow of these modules.)

- Hispanic Name Preprocessor (HNP),
- Hispanic Search Engine (HSE), and
- Hispanic Filter and Sorter (HFS).

3.2. HISPANIC NAME PREPROCESSOR MODULE DECOMPOSITION

3.2.1. **Identification**

This module is known as the Hispanic Name Preprocessor (HNP).

3.2.2. **Type**

The HNP is a subprogram of the HNA-E program that accepts input from the Advanced Name Classifier for CLASS-E (ANC-E) and prepares it for handling by the Hispanic Search Engine (HSE). (See Section 3.13.)

3.2.3. **Purpose**

Hispanic names account for almost 50% of the VLDB name records. In addition to the volume of occurrence, there are many names that occur very frequently. The format of Hispanic names contributes further obstacles to name searching: the surname generally consists of two names and the given names generally consists of two names. The most highly frequently occurring prefix in the VLDB is also Hispanic: DE. The frequency, density and the nature of the name argue for preparing the name in whatever way(s) are necessary to expedite the retrieval process. That is the function of the HNP.

3.2.4. **Function**

3.2.4.1. The HNP will prepare a name identified as Hispanic by the ANC-E for the HSE by

- identifying name segments and determining their disposition,
- manipulating the name segments to generate additional query formats,
- determining name length and record gender,
- specifying the frequency character of each name segment and
- generating search keys.

3.2.4.2. Because of the significant amount of information that is to be generated and collected about the name through the HNP, it is strongly recommended that the name be treated as an object that "knows" what sorts of information it needs. Such an object will provide a mechanism for following the acquisition of information as the object passes through the system. Much of that information will be collected and loaded during the HNP stage.

3.2.5. **Subordinates**

The HNP has ten subordinate functions:

- Name Length Determiner (NLD)
- Hispanic Surname Segmenter (HSS)
- Hispanic TAQ Processor (HTP)
- Hispanic Segment Positioner (HSP)
- Segment Position Identifier (SPI)
- Hispanic Name Formatter (HNF)
- Hispanic Gender Identifier (HGI)

- Frequency Path Director (FPD)
- High Frequency Name Processor (HFP)
- Low Frequency Name Processor (LFP)

3.3. NAME LENGTH DETERMINER MODULE DECOMPOSITION

3.3.1. Identification

This function is known as the Name Length Determiner (NLD).

3.3.2. Type

The NLD is a function that accepts as input a surname (SN) segment and stores the surname length. The length will be used by the Hispanic Surname Segmenter (Section 3.4).

3.3.3. Purpose

Name segment length will provide information that will be used by the Hispanic Surname Segmenter to attempt to divide surnames over a specific length into component segments.

3.3.4. Function

3.3.4.1. The NLD will accept as input each SN segment.

3.3.4.1.1. A segment is a string of characters surrounded by white space.

3.3.4.1.2. The NLD will count the number of characters in SN segment (not including surrounding blanks).

3.3.4.1.3. The NLD will store the length count associated with each SN segment.

3.3.5. Subordinates

None.

3.4. HISPANIC SURNAME SEGMENTER MODULE DECOMPOSITION

3.4.1. Identification

This function is known as the Hispanic Surname Segmenter (HSS).

3.4.2. Type

The HSS attempts to divide surnames over a specified length into component segments. The HSS is a function that must follow the NLD and precede the Hispanic TAQ Processor (Section 3.5).

3.4.3. Purpose

Hispanic names often have many segments and these segments may be quite long. Field lengths of fixed size may not be able to accommodate the number of name segments that occur. Data entry operators often attempt to reduce the name length by conjoining name segments. Conjoined segments have an especially negative impact on the surname. The access point into the database is through the surname and conjoined name segments generally make the component segments inaccessible to processing. Separating conjoined surnames would, therefore, improve the search process.

3.4.4. **Function**

3.4.4.1. The HSS will separate conjoined HF SN segments from a surname segment of nine characters or more in length.

3.4.4.1.1. The HSS will generate additional query records for the separated SN segment and tag the items separated.

3.4.4.1.2. The HSS will generate alias record adds for the separated SN name segments.

3.4.4.2. The HSS will access the High Frequency Surname Type Data Store (HFST).

3.4.4.2.1. Phase 1: The HSS will begin with the leftmost character of the query/add SN segment and attempt to identify a HFSN_TYPE within the input SN string.

3.4.4.2.2. The HSS will choose the longest HFSN_TYPE that it can identify, separate that string from the input string and proceed to Phase 2.

3.4.4.2.3. Phase 2: The HSS will begin with the rightmost character of the query/add SN segment and attempt to identify a HFSN_TYPE (in reverse order) within the remaining input string (after any HFSN_TYPE has been removed during Phase 1).

3.4.4.2.4. The HSS will choose the longest HFSN_TYPE that it can identify and separate that string from the remaining input string.

3.4.4.2.5. Any residual segment will be retained as is.

3.4.4.2.6. If no HFSN_TYPE can be identified in either Phase, no action will be taken.

3.4.4.2.7. An alias (or additional query) will be generated for the divided string.

Figure 1: Example: Hispanic Surname Segmenter

INPUT NAME	HFSN_TYPE	PHASE 1	PHASE 2	OUTPUT
GARCIAGOMEZ	GARCIA GOMEZ	GARCIA	GOMEZ	GARCIA GOMEZ
PEREZDELOPEZ	PEREZ LOPEZ	PEREZ	DELOPEZ	PEREZ DE LOPEZ
BOMEZDEPEREZ	PEREZ	BOMEZDE	PEREZ	BOMEZDE PEREZ
RAMIREZDELAPAZ	RAMIREZ PAZ	RAMIREZ	DELAPAZ	RAMIREZ DELA PAZ

3.4.5. **Subordinates**

None.

3.5. HISPANIC TITLE/AFFIX/QUALIFIER (TAQ) PROCESSOR MODULE DECOMPOSITION

3.5.1. **Identification**

This module will be known as the Hispanic Title/Affix/Qualifier Processor (HTP).

3.5.2. **Type**

The HTP is a process that accepts a full Surname (SN) or Given Name (GN), accesses the Hispanic TAQ Data Store and reduces name fields with multiple segments to their name stems.

3.5.3. **Purpose**

Hispanic names frequently contain peripheral name elements, such as DE, DE LA, DEL, SAN. Matching on these segments is not generally useful; the name segments with information value are the name stems. For example, GARCIA is the more valuable segment in the string DE GARCIA, as is ANGELES in DE LOS ANGELES. Removal of or disregard for the peripheral name elements allows more emphasis to be placed on the name stems, thus improving the search process.

3.5.4. **Function**

3.5.4.1. The HTP will access the Hispanic TAQ Data Store (HTD) to identify TAQ segments: titles (e.g., SR., MR.), affixes (e.g., DE) or qualifiers (e.g., PH.D., HIJO).

3.5.4.1.1. The HTD will contain information about the disposition of the TAQ.

3.5.4.1.2. The HTD will contain information about the type of TAQ (TAQ_TYPE): Title, Prefix, Infix, Suffix, Qualifier.

3.5.4.2. The HTP will scan all SN segments or all GN segments for any TAQ segments.

3.5.4.2.1. The HTP will begin with the leftmost character of the SN or GN field and attempt to identify a TAQ segment among the SN segments and among the GN segments. (The TAQ segment will be surrounded by white space.)

3.5.4.2.2. If the HTP identifies a segment, it will tag the segment with the ID_NO and disposition, as indicated in the HTD.

3.5.4.2.3. If the following segment is also a TAQ segment, it will tag the segment with the ID_NO and disposition, as indicated in the HTD.

3.5.4.2.4. This will continue until all *consecutive* TAQ segments have been tagged.

3.5.4.2.5. When the HTP encounters a following segment that is not a TAQ segment, it will treat that segment as a stem.

3.5.4.2.5.1. Each TAQ segment identified up to that point will be given the TAQ_TYPE P (prefix) and each will be associated and stored with the following stem.

3.5.4.2.6. The HTP will move to the next segment following the stem and will repeat the TAQ identification process.

3.5.4.2.6.1. The HTP will tag all TAQ segments with the ID_NO and disposition.

3.5.4.2.6.2. When the HTP encounters a stem, it will tag each TAQ segment (not yet associated with a stem) with the TAQ_TYPE P and will associate and store each TAQ segment with the following stem.

3.5.4.2.7. If HTP encounters a TAQ segment or segments that has no following stem, it will access the HTD to determine if the TAQ type is a Suffix (S).

3.5.4.2.7.1. If the TAQ has a TAQ_TYPE S, the TAQ will be associated and stored with the *preceding* stem.

3.5.4.2.7.2. The preceding stem may already have prefixal TAQs.

3.5.4.2.7.3. If the TAQ type is not equal to S, the TAQ will be tagged a Stranded Prefix.

3.5.4.3. The HTP will process any TAQ segments identified according to the treatment indicated in the HTD.

3.5.4.4. Treatment options include DELETE, DISREGARD and REMOVE.

3.5.4.4.1. **DELETE** means that the segment is completely disregarded in the remainder of the name search process and contributes marginal information to the filtering process. (N.B. The segment is not deleted from the record.)

3.5.4.4.2. **DISREGARD** means that the segment is disregarded in the remainder of the name search process but contributes to the evaluation of the name in the filtering processes.

3.5.4.4.3. **REMOVE** means that a segment that is conjoined to the name stem is removed from that stem. It is then submitted to additional handling, either DELETE or DISREGARD.

3.5.4.4.3.1. The HTP will begin with the leftmost character in the input stem segment (after free-standing TAQs have been removed) and will attempt to identify all TAQ segments that have been marked for removal (REMOVE).

3.5.4.4.3.2. The HTP will begin with the longest TAQ segment and attempt to remove that; it will then proceed to shorter segments.

3.5.4.4.3.3. If the segment that is to remain after TAQ removal is two characters or fewer, the HTP will not remove the TAQ.

3.5.4.4.3.4. If the TAQ segment is identified and the residual stem is of sufficient length, it is separated from the stem.

3.5.4.4.3.5. The HTP assigns and stores the ID_NO of the removed TAQ.

3.5.4.4.3.6. The HTP then submits the removed TAQ to the treatment indicated (DELETE or DISREGARD) in the HTD and tags and stores the TAQ with that treatment indicator.

Figure 2: Example: TAQ REMOVE Process

INPUT STRING	TAQ REMOVE	OUTPUT
DECORDOBA	DE	DE CORDOBA
DELOSANGELES	DELOS	DELOS ANGELES
DEARING	DE	DE ARING
MARIADE	DE	MARIADE
DELPILAR	DEL	DEL PILAR

3.5.4.5. After all TAQ segments have been submitted to the appropriate process, the remaining segments will be considered SN and GN stems.

Figure 3: Example: TAQ Processing

INPUT SN:	TAQs and STEMS	OUTPUT SN:
DE	TAQ: DE (ID_NO, REMOVE, DISREGARD)	
LA	TAQ: LA (ID_NO, REMOVE, DISREGARD)	
CRUZ	STEM: CRUZ	CRUZ
DE	TAQ: DE (ID_NO, REMOVE, DISREGARD)	
BARRIOS	STEM: BARRIOS	BARRIOS
SAN	TAQ: SAN (ID_NO, DISREGARD, Stranded Prefix)	

3.5.5. **Subordinates**

None.

3.6. HISPANIC SEGMENT POSITIONER MODULE DECOMPOSITION

3.6.1. Identification

This function is known as the Hispanic Segment Positioner (HSP).

3.6.2. Type

The HSP is a function that moves a high frequency (HF) surname (SN) found in the given name (GN) field into the SN field.

3.6.3. Purpose

Surnames that occur in the GN field deprive the match process of relevant SN information. Moving a SN segment that occurs in the GN field to the SN field will benefit the search process. (The SN segment is moved to the rightmost position to retain the value assigned to the resident SN segment(s).)

3.6.4. Function

3.6.4.1. If more than one GN segment (stem) occurs in the GN field, the HSP will determine if the final (rightmost) segment in the GN string is a HF SN.

3.6.4.2. The HSP will move the segment to the SN field.

3.6.4.2.1. If more than one GN segment occurs in the GN field, the HSP will access the High Frequency Surname Type Data Store (HFST) to determine if the rightmost GN segment is a HFSN_TYPE.

3.6.4.2.2. If the segment is a HFSN_TYPE, the HSP will move the segment into the rightmost position of the SN field.

3.6.4.3. The process applies to one name segment only and is not iterative.

Figure 4: Example: Hispanic Segment Positioner (HSP)

INPUT NAME	HFSN_TYPE	OUTPUT FORMAT
CASTRO, MARIA LUZ GOMEZ	GOMEZ	CASTRO **GOMEZ**, MARIA LUZ
BARRIOS LUNA, JUAN PEREZ	PEREZ	BARRIOS LUNA **PEREZ**, JUAN
LOPES ARRIAGA, CARLOS VITRAL	....	LOPES ARRIAGA, CARLOS **VITRAL**

3.6.4.4. An additional query record is generated with the moved segment; the original record is not changed.

3.6.4.5. An alias record add is generated with the moved segment; the original record is not changed.

3.6.5. Subordinates

None.

3.7. HISPANIC NAME FORMATTER MODULE DECOMPOSITION

3.7.1. Identification

This module is known as the Hispanic Name Formatter (HNF).

3.7.2. **Type**

3.7.2.1. The HNF is a process that generates additional name formats for input records that have more than two surname stems.

3.7.2.2. The HNF will follow the HSS, HTP, and HSP.

3.7.2.3. The generated formats will serve as the name format for HF name processing and for comparison in the filtering and sorting process.

3.7.3. **Purpose**

The HNF will limit the number of segments that can occur in the surname field to two in order to maximize the efficient processing of the input name.

3.7.4. **Function**

3.7.5. The HNF will generate additional alias record adds and queries for surnames that contain more than two SN stems.

3.7.5.1. The HNF will accept input strings with any number of SN segments (stems).

3.7.5.2. When more than two SN segments are present, the HNF will generate additional name formats with a limit of two SN segments.

3.7.5.2.1. The HNF will begin with the leftmost SN segment and generate dual-SN formats with each additional SN segment.

3.7.5.2.2. The HNF will move to the second SN segment and generate dual-SN formats with each other SN segment that have not yet been generated.

3.7.5.2.3. The relative order of all segments will be maintained.

3.7.5.3. All generated formats will be stored with the record add.

3.7.5.4. All generated formats will be additional queries.

Figure 5: Example: Hispanic Name Formatter (HNF)

INPUT SURNAME	**GARCIA**	**LUNA**	**BUSTOS**	**ARRIAGA**
HNF DUAL-SN FORMATS	GARCIA	LUNA		
	GARCIA		BUSTOS	
	GARCIA			ARRIAGA
		LUNA	BUSTOS	
		LUNA		ARRIAGA
			BUSTOS	ARRIAGA

3.7.6. **Subordinates**

None.

3.8. SEGMENT POSITION IDENTIFIER MODULE DECOMPOSITION

3.8.1. Identification

This module is known as the Segment Position Identifier (SPI).

3.8.2. Type

The SPI is a function that identifies the relative position of each of the SN and GN stems. The SPI must follow the HTP, HSP and HNF. Segment position information will be accessed by the High Frequency Processor (HFP) and the Hispanic Filter and Sorter (HFS).

3.8.3. Purpose

Hispanic names generally contain more than one SN and more than one GN. The value of each of these name stems is different. In a SN, the leftmost stem is the family name; other SN stems are differentiators. The family name carries more value in the SN. In a GN, the leftmost name stem generally indicates gender so is a valuable indicator. Names that are in- and out-of position are therefore of differing relevance. Position information can contribute to the selection and evaluation of relevant records.

3.8.4. Function

3.8.4.1. The SPI will operate on any SN or GN except where dual-SN formats have been generated.

3.8.4.1.1. Where dual-SN formats have been created, the SPI will accept only those formats.

3.8.4.1.2. The SPI will accept any number of GN segments.

3.8.4.2. The SPI will specify the position in the name field (SN or GN fields) of each name segment.

3.8.4.3. The SPI will begin with the leftmost segment and assign Position 1, proceeding to the next segment and assign Position 2, and so forth.

3.8.4.4. Position information will be generated for and stored with each SN segment.

3.8.4.5. Position information will be generated for and stored with each GN segment.

3.8.5. Subordinates

None.

3.9. HISPANIC GENDER IDENTIFIER MODULE DECOMPOSITION

3.9.1. Identification

This function is known as the Hispanic Gender Identifier (HGI).

3.9.2. Type

This is a function that associates a gender value with a **record**; it will be accessed by the Hispanic Decision Matrix (HDM) and the Hispanic Filter and Sorter (HFS).

3.9.3. Purpose

It is usually possible to predict the gender of a Hispanic name based on the gender marker of the leftmost given name segment. Because crossed-gender names are of little value in the visa adjudication process, lowering the value of a record whose gender does not match that of a query would improve the name matching process.

Predicting gender based on one source of gender, however, may result in elimination of records that differ by one character only. More than one source of gender information can provide a means of validating the gender assignment. This will be the record gender. Record gender will reduce the chance of qualifying or disqualifying a record based on the gender of a single name segment, which could be misspelled or ambiguous with respect to gender.

3.9.4. Function

3.9.4.1. The HGI will derive a gender that will be associated with a *record* and not a Given Name stem alone.

3.9.4.2. A record gender value may be Male (M), Female (F), or Unknown/Ambiguous (U) Gender.

3.9.4.2.1. The HGI will derive the record gender from the GN gender associated with each GN segment and the gender provided by the user during the data entry process.

3.9.4.2.1.1. The HGI will access the Hispanic Given Name Type Data Store (HGT) to determine the gender associated with each GN segment.

3.9.4.2.1.1.1. If the name is present in the HGT, the gender indicated will be associated with the GN segment.

3.9.4.2.1.1.2. If the name is not present in the HGT, the record gender will be marked as Unknown (U). (This would occur for a query with a name never before submitted to the system.)

3.9.4.2.1.2. The applicant gender is determined at the time of application and must be entered, captured and stored by the system.

3.9.4.2.2. The HGI will verify that all gender indicators agree: the gender associated with each GN segment and the applicant gender received at the time of application.

3.9.4.2.2.1. To mark the record gender as M or F, the HGI requires gender validation from a ***minimum*** of two sources.

3.9.4.2.2.2. All sources of gender information (whether two or more) must match for gender to be marked as M or F.

3.9.4.2.2.2.1. If the gender indicators match, the match value will become the record gender.

3.9.4.2.2.2.2. If the gender indicators do not match, gender is marked as U.

Figure 6: Example: Record Gender Assignment

GIVEN NAME	HGT GNDR	INPUT GENDER	RECORD GENDER
1) MARIA	F	F	F
LUZ	F		
2) JOSE	M	M	M
ANTONIO	M		
3) CARLOS	M	M	U
(DE LA) CRUZ	U		
4) BERNARDO	M	M	M
5) CAMEN (misspelling)	(not in HGT)	F	U
MARIA	F		

3.9.5. **Subordinates**

None.

3.10. FREQUENCY PATH DIRECTOR MODULE DESCRIPTION

3.10.1. **Identification**

This module is known as the Frequency Path Director (FPD).

3.10.2. **Type**

3.10.2.1. The FPD directs a record to the High Frequency Processor or Low Frequency Processor depending on the presence or absence of HF *surnames* in the string.

3.10.2.2. The FPD will access the following data stores:

- High Frequency Surname Type Data Store (HFST)
- Hispanic Character Data Store (HCD)

3.10.3. **Purpose**

Many Hispanic names occur with such high frequency that they would benefit from special processing. The system must determine which the high frequency surnames are and direct records with high frequency surnames to the proper handler.

3.10.4. **Function**

3.10.4.1. The FPD will accept any SN format, except where dual-SN formats have been generated by the Hispanic Name Formatter (HNF).

3.10.4.1.1. The FPD will operate on the dual-SN formats where they have been generated.

3.10.4.2. The FPD will identify, process and assign keys to SN initials.

3.10.4.3. The FPD will identify and tag each SN stem as HF or LF.

3.10.4.4. The FPD will assign HFSN_KEYs, where appropriate.

3.10.4.5. The FPD will direct the record to the HF Processor or LF Processor depending on the frequency tags of the SN segments.

3.10.4.6. The frequency-identification process will repeat until the frequency value of all SN segments has been identified.

3.10.4.7. **Surname Initials**

3.10.4.8. **Record Adds**

3.10.4.9. The FPD will generate a SN_INIT Key for the initial character of each SN segment (The SN segment may be an initial).

3.10.4.9.1. The FPD will access the Hispanic Character Data Store (HCD) to identify the SN_INIT Key.

3.10.4.9.2. The FPD will find the initial character in the CHAR list.

3.10.4.9.3. The FPD will assign the SN_INIT Key to the character.

3.10.4.9.4. The SN_INIT Key will be the SET_ID for all occurrences of the character.

3.10.4.9.5. The FPD will store the SN_INIT Key with the SN segment of the record.

3.10.4.10. **Query**

3.10.4.11. The FPD will identify single characters that occur in the SN field; any segment that has a name length of 1 (as specified by the Name Length Determiner (NLD)) is an initial.

3.10.4.12. The FPD will access the Hispanic Character Data Store (HCD) to determine the SN_INIT Key(s) to assign to the segment.

3.10.4.12.1. The FPD will find each instance of the character in the CHAR_VAR list.

3.10.4.12.2. The FPD will assign SN_INIT Key(s) to the SN initial.

3.10.4.12.3. The SN_INIT KEY is the SET_ID associated with each instance of the initial.

3.10.4.12.4. The SN initial may have multiple SN_INIT Keys.

3.10.4.13. The FPD will ignore the SN_INIT Keys when determining the frequency path assignment of a record; the assignment will be based on the frequency of the other SN segment.

3.10.4.14. **High Frequency Surnames**

3.10.4.15. The FPD will access the High Frequency Surname Type Data Store (HFST).

3.10.4.16. If a SN segment matches exactly a HFSN_TYPE in the HFST, the segment will be given the HFSN_KEY associated with the HFSN_TYPE.

3.10.4.16.1. **Record Add/Query**: The HFSN_KEY will be the SET_ID associated with the HFSN_TYPE in the HFST.

3.10.4.16.2. **Record Add**: A digraph value (DI_VAL) of 1.00 will be assigned to and stored with the segment that matches a HFSN_TYPE exactly.

3.10.4.16.3. The HFSN_KEY will represent a set of name segments that have qualified as digraph variants of the HFSN_TYPE. (See 3.12.4.38 for information on how the variants are assigned to the same set.)

3.10.4.17. The FPD will direct records that contain *all* HFSN_KEYs to the High Frequency Processor (HFP).

Figure 7: Example: Qualification for High Frequency Processor

INPUT NAME: GARCIA LOPEZ, ANTONIO JESUS	FIELD	HFSN_KEY	DI_VAL
GARCIA	SN	0001	1.00
LOPEZ	SN	0004	1.00

Figure 8: Resource: Piece of High Frequency Surname Type Data Store (HFST)

ID_NO	HFSN_TYPE	SET_ID
0001	GARCIA	0001
0002	RODRIGUEZ	0002
0003	HERNANDEZ	0003
0004	LOPEZ	0004
0005	MARTINEZ	0005
0006	GONZALEZ	0006
0007	PEREZ	0007
0008	SANCHEZ	0008
0009	RAMIREZ	0009
0010	GOMEZ	0010
0011	...	0011

3.10.4.18. The FPD will direct all records that do *not* contain all HFSN_KEYs to the Low Frequency Processor (LFP).

3.10.5. **Subordinates**

None.

3.11. HIGH FREQUENCY PROCESSOR MODULE DECOMPOSITION

3.11.1. **Identification**

This module is known as the High Frequency Processor (HFP).

3.11.2. **Type**

3.11.2.1. The HFP is a program module that

- will process records with all HFSN_KEYs, HFSN_VAR Keys, SN_INIT Keys and mixed HF and LF Keys;
- will generate Given Name Keys; and
- will access the Hispanic Decision Matrix Data Store (HDM) to identify retrieval criteria for the HF records.

3.11.2.2. Multiple entry points into the HFP will be supported: through the Frequency Path Director and the Low Frequency Processor (LFP).

3.11.3. **Purpose**

Earlier attempts to develop a HF handler for Hispanic names have been limited to processing of records that contain only HF names; little to no variation was permitted. HNA-E will support variation in the processing of HF names by allowing multiple entry points into the HFP.

3.11.4. **Function**

3.11.4.1. The HFP will accept names directed to the processor by the Frequency Path Director (FPD) and by the LFP.

3.11.4.2. The records accepted will contain all HFSN_KEYs and/or HFSN_VAR Keys, SN_INIT Keys, and mixed HFSN_KEY/HFSN_VAR and DI_KEYs.

3.11.4.3. All SN_INIT Keys passed to the HFP will be treated as segment Keys and will undergo the same criteria identification as other segments.

3.11.4.4. If all segments of the SN Field have been given HFSN_KEYs and/or HFSN_VAR Keys and related DI_VALs, the HFP will begin processing the GN segments.

3.11.4.5. **Processing the Given Name Segments**

3.11.4.6. If the GN segment is First Name Unknown (FNU), no GN processing will take place.

3.11.4.7. **High Frequency Given Name Segment Keys**

3.11.4.8. **Record Adds**

3.11.4.9. The HFP will access the Hispanic Given Name Variant Data Store (HGNV) to determine if the GN segments are HF GN segments.

3.11.4.9.1. If the GN segment matches one or more variants in the HGNV, the HFP will assign the HFGN_KEY to the GN segment.

3.11.4.9.2. The HFGN_KEY is(are) the SET_ID(s) associated with the variant.

3.11.4.9.3. The HFP will associate the appropriate DI_VAL with the SET_ID and GN segment.

3.11.4.9.4. The HFP will store the SET_ID(s) and their DI_VAL with the GN segment.

3.11.4.9.5. The HFGN_KEY ensures that the system will retrieve variants of a HF segment when the HF segment is queried.

3.11.4.10. **Query**

3.11.4.11. The HFP will access the Hispanic Given Name Type Data Store (HGT).

3.11.4.11.1. If a GN segment matches exactly a GN_TYPE name segment in the HGT and HI_FREQ = 1 (is True) (that is, the segment is a HF GN_TYPE segment), the HFP will assign to the GN segment the HFGN_KEY associated with the GN_TYPE.

3.11.4.11.2. The HFGN_KEY will be the SET_ID associated with the HF GN_TYPE.

3.11.4.12. **High Frequency Given Name Initial Keys**

3.11.4.13. The HFP will create one or more GN_INIT Keys for each HF GN segment.

3.11.4.14. The GN_INIT Key will be the initial key for each GN segment, including initials.

3.11.4.14.1. **Record Add**

3.11.4.14.2. The HFP will identify the initial character of each GN segment.

3.11.4.14.3. The HFP will access the Hispanic Character Data Store (HCD) and will find all occurrences of the character in the CHAR-VAR list.

3.11.4.14.4. The HFP will assign the GN_INIT Key(s) to each GN initial.

3.11.4.14.4.1. The GN_INIT Key will be the SET_ID(s) associated with the GN initial (CHAR_VAR).

3.11.4.14.4.2. The GN segment may have multiple GN_INIT Keys.

3.11.4.14.4.3. The GN_INIT Key will permit retrieval of multiple initials for a GN initial.

3.11.4.14.4.4. The HFP will store the GN_INIT Key(s) for each GN segment initial with the record.

3.11.4.14.5. Query

3.11.4.14.6. The HFP will access the HCD and find the initial in the CHAR list.

3.11.4.14.7. The HFP will identify the GN_INIT Key for each GN segment initial.

3.11.4.15. If the GN segment is not a variant in the HGNV, the HFP will tag the name as LF.

3.11.4.16. **Low Frequency Given Name Segment Keys**

3.11.4.17. For each LF GN segment, the HFP will attempt to determine if the segment is a potential variant of a HF GN_TYPE and will create one or more GN_INIT Keys for record adds and queries.

3.11.4.18. **Record Add**

3.11.4.19. If the LF GN is *not* in the HGNV Data Store, the HFP will determine if the segment is a potential variant of a HFGN_TYPE. (This would apply to LF GN segments that are being submitted to the system for the first time.)

3.11.4.19.1. If the HFP determines that the HF GN segment is a variant of a HFGN_TYPE, the LFP will append the segment to the HFGV Data Store.

3.11.4.19.2. The HFP will access the Hispanic Given Name Type Data Store (HGT) to determine if the LF GN segment is a digraph variant of one or more of the HF GN_TYPEs. (That is, the LF GN segment is a digraph variant of the GN_TYPE whose HF Value is True (1)).

3.11.4.19.3. The LFP will perform a digraph evaluation of the LF GN and each HF GN_TYPE.

3.11.4.19.4. The digraph value is determined in the following way:

3.11.4.19.4.1. The digraphs are identified for each segment.

3.11.4.19.4.2. Each pair of alphabetic characters is identified: CARA → CA / AR / RA

3.11.4.19.4.3. A digraph is also formed of the initial boundary (#) and the first alphabetic character: CARA → #C.

3.11.4.19.4.4. A digraph is also formed of the final alphabetic character and the final boundary (#): CARA → A#.

3.11.4.19.4.5. The number of shared digraphs is calculated.

3.11.4.19.4.5.1. A digraph may match one digraph only.

3.11.4.19.4.6. The number of shared digraphs is multiplied by 2 and divided by the total number of digraphs in Comparand #1 added to the total number of digraphs in Comparand #2.

3.11.4.19.4.6.1. The formula is:

$2 * d / a + b$,
where d = the total number of shared digraphs;
where a = the total number of digraphs in Comparand #1; and
where b = the total number of digraphs in Comparand #2.

3.11.4.19.4.7. The result is the Digraph Value (DI_VAL) for the two Comparands.

Figure 9: Example: Digraph Calculation

COMPARANDS	DIGRAPHS	SHARED DIGRAPHS	DI_VAL
COMPARAND #1: CARA	#C CA AR RA A# (5 total digraphs = a)	#C CA AR A#	2*d / a + b = 8 / 12
COMPARAND #2: CARINA	#C CA AR RI IN NA A# (7 total digraphs = b)	= 4 (d)	**0.67**

3.11.4.19.4.8. This process is performed for each of pair of Comparands.

3.11.4.19.5. To qualify for addition to the HFGV as a variant of one or more HFGN_TYPEs, the digraph value must pass a threshold, the High Frequency Given Name Variant Threshold (HFGV Threshold).

3.11.4.19.5.1. The HFP will access the Hispanic Parameter Data Store (HPD) (Section 4.13) to determine the HFGV Threshold that the digraph value must pass for the LF GN to be appended to the HFGV Data Store.

3.11.4.19.5.2. If the LF GN segment qualifies as digraph variant of one or more HF GN_TYPEs, the HFP

- will append the LF GN to the HFGN_TYPEs to which it is related by entering the name into the HFGN_VAR list in the HFGV Data Store;
- will assign the next available ID_NO to the newly added HFGN_VAR;
- will assign the SET_ID to the newly added HFGN_VAR that corresponds to the SET_ID of the HFGN_TYPE with which the new HFGN_VAR is associated;
- will enter the digraph value into DI_VAL; and
- will store with the LF GN segment in the record the ID_NO(s) of the HFGN_VAR for each entry, the SET_ID of each HFGN_TYPE that is the parent of the HFGN_VAR, and the digraph value associated with each entry.

3.11.4.20. Whether or not the LF segment is a variant of a HFGN_TYPE, the HFP will generate one or more GN_INIT Keys for the LF GN segment.

3.11.4.20.1. The GN_INIT Key will be the initial key for each GN segment, including segments that are initials.

3.11.4.20.2. If the GN segment is FNU (First Name Unknown), no GN_INIT Key will be generated.

3.11.4.20.3. **Record Add**

3.11.4.20.4. The HFP will identify the initial character of each GN segment.

3.11.4.20.5. The HFP will access the Hispanic Character Data Store (HCD) and will find all occurrences of the character in the CHAR-VAR list.

3.11.4.20.6. The HFP will assign the GN_INIT Key(s) to each GN initial.

3.11.4.20.6.1. The GN_INIT Key will be the SET_ID(s) associated with the GN initial (CHAR_VAR).

3.11.4.20.6.2. The GN segment may have multiple GN_INIT Keys.

3.11.4.20.6.3. The GN_INIT Key will permit retrieval of multiple initials for a GN initial.

3.11.4.20.6.4. The HFP will store the GN_INIT Key(s) for each GN segment initial with the record.

3.11.4.20.7. Query

3.11.4.20.8. The HFP will access the HCD and find the initial in the CHAR list.

3.11.4.20.9. The HFP will identify the GN_INIT Key for each GN segment initial.

3.11.4.20.10. The GN_INIT Key will be the SET_ID associated with the CHAR.

Figure 10: Example: HFGN_KEYs and GN_INIT Keys (Query)

INPUT GN	HF?	HFGN_KEY	GN_INIT KEYS
MARIO	T	020	078 (M)
MICHAEL	F		078 (M)
YSABEL	F		036 (Y, I)
ZUSANA	F		002 (Z, S)

Figure 11: Example: HFGN_KEYs and GN_INIT Keys (Record Add)

INPUT GN	HF?	HF VARIANT?	HFGN_TYPE	HFGN_KEY	GN_INIT KEYs
MARIO	T		MARIO	020	078 (M)
MICHAEL	F	F			078 (M)
YSABEL	F	T	ISABEL	203	036 (Y, I)
ZUSANA	F	T	SUSANA	436	002 (Z, S)

3.11.4.21. The HFP will direct queries with all HF SN or mixed HF and LF SN (including SN_INIT Keys) and any GN Keys (HFGN_KEYs or GN_INIT Keys) or FNU to the Hispanic Decision Matrix (HDM) to determine the record retrieval criteria.

3.11.4.21.1. Criteria for database retrieval include name content (whether the names are the same or different), the position of the name segments, the YOB range, the Refusal Code Level, Record Gender and additional restrictions based on the GN.

3.11.4.22. **Hispanic Decision Matrix**

3.11.4.23. The HFP will access the portion of the HDM that represents the number of HF SN segments in the query name, either one HF SN or two HF SNs.

3.11.4.24. The HFP will identify and generate the set of SN formats possible for the number of SN segments in the query (one or two).

3.11.4.24.1. The SN formats indicate

- position of segments,
- number of segments, and
- other segments permitted.

3.11.4.25. The HFP will identify the retrieval criteria in the HDM associated with each SN format.

3.11.4.25.1. The retrieval criteria include

- Year-of-Birth Range
- Refusal Level and
- Record Gender

3.11.4.26. GN Keys will be carried forward with the retrieval criteria.

3.11.4.27. The HFP will send to the Hispanic Search Engine (HSE) the query format(s), all retrieval criteria associated with each query format and all SN Keys and all GN Keys generated for the query.

Figure 12: Example: Hispanic Decision Matrix (Values for example only)

	Single-Segment SN				**Two-Segment SN**							
QUERY SN FORMAT	A	A	A		AB	AB	AB	AB	AB	AB	AB	AB
DATABASE SN FORMATS	A	AB	BA		AB	BA	A	B	AC	CA	CB	BC
YR	5	5	2		5	4	4	2	2	0	0	0
RL	4	4	3		4	4	4	1	1	0	0	0
RGNDR	MFU	MFU	MFU		MFU	MFU	MFU	MFU	FU	MFU	MFU	MFU

3.11.5. **Subordinates**

None.

3.12. LOW FREQUENCY PROCESSOR MODULE DECOMPOSITION

3.12.1. Identification

This module is known as the Low Frequency Processor (LFP).

3.12.2. Type

3.12.2.1. The LFP is a program module that will process names that contain SN segments identified by the FPD as Low Frequency SN segments (i.e., not found in the HFST Data Stores by the FPD).

3.12.2.2. The LFP will access the

- High Frequency Surname Variant Data Store (HFSV) and
- Low Frequency Surname Type Data Store (LFST).

3.12.3. Purpose

The LFP will process name segments that are identified as LF SN segments by the FPD. The LFP will determine 1) whether or not the LF segment is a variant of one or more HF SN and 2) whether or not the LF segment has variants among the LF segments listed in the Low Frequency Surname Type Data Store (LFST). The result of these two processes will be a list of segments to use as exact matches for retrieval.

3.12.4. Function

3.12.4.1. General

3.12.4.2. The LFP will accept from the FPD any record with a SN segment that has been tagged as a LF SN.

3.12.4.2.1. The LFP will process records that contain only LF segments *and* all records that contain mixed HF and LF segments.

3.12.4.2.2. The LFP will process records that contain one LF segment and SN_INIT Keys.

3.12.4.2.2.1. Low frequency processing will be limited to the LF segment.

3.12.4.2.2.2. The SN_INIT Keys will contribute to the building of LF retrieval keys.

3.12.4.2.3. With mixed HF and LF SN, the LFP will process only the LF SN segment. (The HF segment will have been assigned a HFSN_KEY by the FPD).

3.12.4.3. *All* LF segments in records that are sent to the LFP will be analyzed for both HF affiliations and LF variants.

3.12.4.4. The LFP will attempt

- to relate each LF SN segment to one or more HFSN_TYPEs,

- to identify other LF SN segments related to the input LF SN segment(s), and
- to append LF SN segments to the HFSV and LFST that have not been previously submitted to the system to the HFSV and LFST.

3.12.4.5. The LFP will direct a query in which all LF SN segment(s) have been related to HFSN_TYPEs to the High Frequency Processor (HFP) (see Section 3.11) for generation of GN Keys and submission to the Hispanic Decision Matrix.

3.12.4.5.1. The record may have the format HFSN_KEY (or SN_INIT Key) + HFSN_VAR Key, where the second key relates a LF SN segment to a HFSN_TYPE.

3.12.4.5.2. The record may have the format HFSN_VAR Key + HFSN_VAR Key, where both segments are keys relating a LF SN segment to a HFSN_TYPE.

3.12.4.5.3. The record may have the format HFSN_VAR Key, where the only segment is a key relating a LF SN segment to a HFSN_TYPE.

3.12.4.6. The LFP will direct a query record in which all LF SN segments have been related to other LFSN_TYPEs directly to the Hispanic Search Engine. (SN_INIT Keys may be present.)

3.12.4.7. The LFP will perform the following processes:

- Access the HFSV Data Store to determine if the LF name segment is variant of a HFSN_TYPE.
- Assign HFSN_VAR Key(s), as appropriate.
- Generate LF_KEYs for LF SN variants identified in the LFS Data Store.
- Perform a digraph comparison on the HFST Data Store to determine if a LF SN not in the HFSV Data Store is a digraph variant of a HFSN_TYPE segment.

3.12.4.8. The goal of the LFP, for a query with LF SN segments, is to develop a set of specific names related to the LF SN that will be used as keys for record retrieval.

3.12.4.9. **Identifying Related High Frequency Surnames**

3.12.4.10. The LFP will access the HFSV Data Store to determine if each LF SN segment in the input name is a variant of HFSN_TYPE.

3.12.4.10.1. The LFP will attempt to find all occurrences of the LF SN segment in the HFSN_VAR list.

3.12.4.10.2. **Record Add**

3.12.4.10.3. If the segment is found in the HFSN_VAR list, the LFP will assign one or more HFSN_KEYs and HFSN_VAR Keys to the LF SN.

3.12.4.10.3.1. The keys will be

- the HFSN_KEY: the SET_ID associated with the HFSN_TYPE that is the parent of the HFSN_VAR and
- the HFSN_VAR Key: the ID_NO of the HFSN_VAR.

3.12.4.10.3.2. The digraph value associated with the HFSN_TYPE and HFSN_VAR pair will be retrieved and stored with the HFSN_KEY and HFSN_VAR Key as the DI_VAL.

3.12.4.10.3.3. The LFP will store the HFSN_KEY, HFSN_VAR Key and the associated digraph value with the record segment.

3.12.4.10.3.3.1. For example, if GARCA is the LF SN and is a variant of the HFSN_TYPE GARCIA, then GARCA will be given the SET_ID associated with the HFSN_TYPE GARCIA (0001) and the ID_NO that uniquely identifies GARCA (000137).

3.12.4.10.3.3.2. The associated digraph value (0.77) will be stored with the LF SN GARCA as the DI_VAL of 0001 and 000137.

3.12.4.10.3.4. There may be multiple HFSN_KEYs and HFSN_VAR Keys associated with a single LF SN segment.

Figure 13: Example: Associating a LF SN Segment with HFSN_TYPE in a Record Add

QUERY SURNAME: PEREZ BOMEZ	**HF SN?**	**HFSN_TYPE**	**HF KEYS**		**DI_VAL**
			HFSN_KEY	**HFSN_VAR KEY**	
PEREZ	T	PEREZ	0007		1.00
BOMEZ	**F**	**GOMEZ**	**0010**	**016978**	**0.67**

Figure 14: Piece of HFST Data Store

ID_NO	HFSN_TYPE	SET_ID
0001	GARCIA	0001
0002	RODRIGUEZ	0002
0003	HERNANDEZ	0003
0004	LOPEZ	0004
0005	MARTINEZ	0005
0006	GONZALEZ	0006
0007	PEREZ	0007
0008	SANCHEZ	0008
0009	RAMIREZ	0009
0010	GOMEZ	0010
0011	...	0011

Figure 15: Piece of HFSV Data Store

ID_NO	HFSN_VAR	SET_ID	DI_VAL
032711	PEREZ	007	1.00
032712	PERES	007	0.67
032713	PEREZA	007	0.77
016976	GOMEZ	010	1.00
016977	GOMES	010	0.67
016978	BOMEZ	010	0.67

3.12.4.10.4. **Query**

3.12.4.10.5. The LFP will attempt to associate the LF SN with one or more HFSN_TYPEs.

3.12.4.10.5.1. The LFP will access the HFSV and determine if the LF SN is a variant of a HFSN_TYPE.

3.12.4.10.5.2. If the LF SN is found in the HFSN_VAR list of the HFSV table, the LFP will assign a HFSN_VAR Key to the LF SN segment.

3.12.4.10.5.2.1. The HFSN_VAR Key will be the ID_NO associated with the HFSN_VAR (and *not* the SET_ID that is associated with the HFSN_TYPE).

3.12.4.10.5.2.1.1. The LF segment will be associated with the HF segment but with no other name segments in the same HFSN_TYPE class.

3.12.4.10.5.2.1.2. That is, the variants associated with the HF segment are *not* related to one another through this process.

3.12.4.10.5.3. A LF SN may be a variant of multiple HFSN_TYPEs and may therefore receive multiple HFGN_VAR Keys.

3.12.4.10.6. If, by virtue of this process, all LF SN segments in a query are set equal to HFSN_VAR Keys, the LFP will direct the query record to the HFP (see Section 3.11) for generation of Given Name Keys, submission to the High Frequency Decision Matrix (HDM) and identification of retrieval criteria.

Figure 16: Example: Associating a LF SN Segment with HFSN_TYPE in a Query

QUERY SURNAME: PEREZ BOMEZ	**HF SN?**	**HFSN_TYPE**	**HF KEYS**	
			HFSN_KEY	**HFSN_VAR KEY**
PEREZ	T	PEREZ	0007	
BOMEZ	**F**	**GOMEZ**		**016978**

Figure 17: Piece of HFST Data Store

ID_NO	HFSN_TYPE	SET_ID
0001	GARCIA	0001
0002	RODRIGUEZ	0002
0003	HERNANDEZ	0003
0004	LOPEZ	0004
0005	MARTINEZ	0005
0006	GONZALEZ	0006
0007	PEREZ	0007
0008	SANCHEZ	0008
0009	RAMIREZ	0009
0010	GOMEZ	0010
0011	...	0011

Figure 18: Piece of HFSV Data Store

ID_NO	HFSN_VAR	SET_ID	DI_VALUE
032711	PEREZ	007	1.00
032712	PERES	007	0.67
032713	PEREZA	007	0.77
016976	GOMEZ	010	1.00
016977	GOMES	010	0.67
016978	BOMEZ	010	0.67

3.12.4.10.7. The LFP will direct all **queries and record adds** to a LF analysis whether or not the LF SN segment was identified as a variant of a HFSN_TYPE.

3.12.4.11. **Identifying Related Low Frequency Surnames**

3.12.4.12. **General**

3.12.4.13. All records with one or more LF SN segments will undergo LF analysis by the LFP.

3.12.4.14. For record adds, the LFP will assign an ID_NO that will be stored with the record.

3.12.4.15. The LFP will generate LFDIKEYs for each LF SN segment in the query.

3.12.4.16. The LFP will use the LFDIKEYs to identify related LF SN segments.

3.12.4.17. Note that the LFP will not generate LF Keys for the GN portion of the input name.

3.12.4.18. **LF SN Segment in LFST**

3.12.4.19. The LFP will determine if the LF SN segment of the input record is in the Low Frequency Surname Type Data Store (LFST).

3.12.4.20. **Record Add**:

3.12.4.21. If the LF SN segment is a LFSN_TYPE in the LFST, the LFP will assign to and store with the LF SN segment the ID_NO associated with the LFSN_TYPE.

3.12.4.22. **Query**:

3.12.4.23. If the LF SN segment is a LFSN_TYPE in the LFST, the LFP will retrieve the (up to) 10 digraph keys (LFDIKEYs) that are associated with the name segment in the LFST.

3.12.4.24. The LFP will use the LFDIKEYs retrieved from the LFST and the LFDIKEYs stored with all LFSN_TYPEs in the LFST Data to subset the LFST and to identify potential variants of the input LF SN segment.

3.12.4.25. **Identifying LF Query Variants**

3.12.4.26. **Phase 1**:

3.12.4.27. The LFP will subset the LFST Data Store.

3.12.4.28. The LFP will select those names from the LFST that share a pre-determined set of LFDIKEYs.

3.12.4.28.1. The LFP will determine the number of LFDIKEYs shared between each LFSN_TYPE and the LF query SN segment.

3.12.4.28.2. The LFP will determine the Shared Key Value based on the number of shared digraphs.

3.12.4.28.2.1. The LFP will use the following formula to determine the Shared Key Value: multiply the number of shared keys by two and divide by the total number of keys associated with each name:

2 * [number of shared keys] / (total keys of Comparand #1 plus total keys of Comparand #2)

3.12.4.28.3. The LFP will select only those LFSN_TYPEs whose Shared Key Value passes the LFDIKEY Threshold.

3.12.4.28.3.1. The LFP will access the Hispanic Parameter Data Store to identify the minimum matching requirement for the Shared Key Value, the LFDIKEY Threshold.

3.12.4.28.3.2. For a segment to qualify for further processing, the Shared Key Value must pass the LFDIKEY Threshold found in the Hispanic Parameter Data Store (HPD).

Figure 19: Example: Phase 1: LF Variants Related to a LF Query SN Segment; LFDIKEY Threshold = 0.40

	ID_NO		LFDIKEYs	SHARED KEYS	PASS LFDIKEY THRESHOLD 0.40?
QUERY NAME #1		FLORENZAN	FL1 / FL2 / LO2 / LO1 / LO3 / OR3 / OR2 / OR4 / RE4 / RE3		
LFSN_TYPE	000189	FLORENZAN	FL1 / FL2 / LO2 / LO1 / LO3 / OR3 / OR2 / OR4 / RE4 / RE3	10 (All)	2*10/20 = 1.00 YES
LFSN_TYPE	000232	FLORESZ	FL1 / FL2 / LO2 / LO1 / LO3 / OR3 / OR2 / OR4 / RE4 / RE3	10 (All)	2*10/20 = 1.00 YES
LFSN_TYPE	000412	LORENZ	LO1 / LO2 / OR2 / OR1 / OR3 / RE3 / RE2 / RE4 / EN4 / EN3	5 (LO2 / OR2 / OR3 / RE3 / RE4)	2*5/20 = 0.50 YES
QUERY NAME #2		TOREAT	TO1 / TO 2 / OR2 / OR1 / OR3 / RE3 / RE2 / RE4 / EA4 / EA3		
LFSN_TYPE	000714	TOREAT	TO1 / TO 2 / OR2 / OR1 / OR3 / RE3 / RE2 / RE4 / EA4 / EA3	10 (All)	2*10/20 = 1.00 YES
LFSN_TYPE	000652	THORET	TH1 / TH 2 / HO2 / HO1 / HO3 / OR3 / OR2 / OR4 / RE4 / RE3	4 (OR2 / OR3 / RE3 / RE4)	2*4/20 = 0.40 YES
LFSN_TYPE	000776	TOERO	TO1 / TO2 / OE2 / OE1 / OE3 / ER3 / ER2 / ER4 / RO4 / RO3	2 (TO1 / TO2)	2*2/20 = 0.20 NO

3.12.4.29. **Phase 2**:

3.12.4.30. The LFP will perform a digraph comparison of each LF query SN segment that passed the LFDIKEY Threshold with each LFSN_TYPE.

3.12.4.31. The digraph comparison will identify the set of names to be retrieved from the database.

3.12.4.31.1. See Section 3.11.4.19.4 for the digraph analysis function and formula.

3.12.4.31.2. The LFP will access the Hispanic Parameter Data Store to determine the LF_DI Threshold.

3.12.4.31.3. For a segment to qualify for further processing, the digraph value must pass the LF_DI Threshold found in the Hispanic Parameter Data Store.

Figure 20: Example: Digraph Filter of LFST Candidate SN Segments

LF QUERY SN: FLORENZAN	LFSN_TYPES PASSING LFDIKEY THRESHOLD	DIGRAPH SCORE	PASS LF_DI THRESHOLD: 0.57?
FLORENZAN	FLORENZAN	2*10/20 = 1.00	YES
FLORENZAN	FLORESZ	2*5/18 = 0.56	NO
FLORENZAN	LORENZ	2*5/17 = 0.59	YES

3.12.4.32. The LFP will assign a key (DI_KEY) to each LFSN_TYPE that passes the LF_DI Threshold.

3.12.4.32.1. The DI_KEY will be the ID_NO associated with the LFSN_TYPE in the LFST.

3.12.4.32.2. The DI_KEY will contribute to the building of the retrieval key.

3.12.4.33. For a segment that passes the LF_DI Threshold, the LFP will retain the digraph score derived from the digraph evaluation and associate it with the appropriate DI_KEY.

3.12.4.34. **Low Frequency Surname Segment *Not* in LFST**

3.12.4.35. **Add:**

3.12.4.36. If the LF SN *record add* segment is **not** in the LFST, the LFP

- will append the LF SN to the LFST as a LFSN_TYPE and assign the next ID_NO available;
- will generate the LFDIKEYs for the new LFSN_TYPE and will add them to the LFST with the LFSN_TYPE;
- will assign the ID_NO to the LF SN segment of added record and
- will determine if the LF SN is a variant of a HFSN_TYPE and therefore should also be added to the HFSV Data Store.

3.12.4.37. The LFP will append the LF SN segment and its LFDIKEYs to the LFST.

3.12.4.37.1. The LFP will assign the next available ID_NO to the newly entered LF SN (LFSN_TYPE).

3.12.4.37.2. The LFP will generate the LFDIKEYs to be associated with the LF SN segment (see 3.13.4.42).

3.12.4.37.3. The up-to-10 keys will be added to the LFST along with the LFSN_TYPE.

3.12.4.37.4. The LFP will assign the LFSN_TYPE ID_NO to the LF SN segment for storage with the record add.

3.12.4.38. The LFP will determine if a LF SN segment that was not identified as a HFSN_VAR in the HFSV *and* that was not identified as a LFSN_TYPE in the LFST is a potential variant of a HFSN_TYPE.

3.12.4.38.1. The LFP will access the High Frequency Surname Type Data Store (HFST) to determine if the LF SN segment is a digraph variant of one or more of the HFSN_TYPEs.

3.12.4.38.1.1. The LFP will perform a digraph evaluation of the LF SN and each HFSN_TYPE. (See Section 3.11.4.19.4 for details of the procedure and formula for performing a digraph evaluation.)

3.12.4.38.1.2. To qualify for addition to the HFSV as a variant of one or more HFSN_TYPEs, the digraph value must pass a threshold, the High Frequency Surname Variant Threshold (HFSV Threshold).

3.12.4.38.1.3. The LFP will access the Hispanic Parameter Data Store to determine the HFSV Threshold that the digraph value must pass for the LF SN to be appended to the HFSV Data Store.

3.12.4.38.2. If the LF SN segment is determined to be a digraph variant of one or more HFSN_TYPEs, the LFP

- will append the LF SN to the HFSN_TYPEs to which it is related by entering the name into the HFSN_VAR list;
- will assign an ID_NO to the newly added HFSN_VAR;
- will assign the SET_ID to the newly added HFSN_VAR that corresponds to the SET_ID of the HFSN_TYPE with which the new HFSN_VAR is associated;
- will enter the digraph value into DI_VAL; and
- will store with the LF SN segment in the record add the ID_NO of the HFSN_VAR for each entry, the SET_ID of each HFSN_TYPE that is the parent of the HFSN_VAR and the DI_VAL for each relationship.

3.12.4.39. **Query**

3.12.4.40. If the LF SN *query* segment is not in the LFST, the LFP

- will generate the LFDIKEYs for the new LF SN;
- will select the related LFSN_TYPEs through the LF selection process; and
- will determine if the LF SN is a variant of a HFSN_TYPE, assign appropriate keys and retain related digraph values. (See Section 3.12.4.43).

3.12.4.41. The LFP will generate the LFDIKEYs for the LF SN segment (see Section 3.12.4.44).

3.12.4.42. The LFP will identify LFSN_TYPEs in the LFST that are variants of the LF. (See Section 3.12.4.11 for the identification process.)

3.12.4.43. The LFP will determine if a LF SN segment that was not identified as a HFSN_VAR *and* that was not found in the LFST is a potential variant of a HFSN_TYPE.

3.12.4.43.1. The LFP will access the HFST Data Store and perform a digraph evaluation between the LF SN and each HFSN_TYPE. (See Section 3.11.4.19.4 for details of the procedure and formula for performing a digraph evaluation.)

3.12.4.43.2. The digraph value must pass a threshold for the LF SN to be considered a variant of a HFSN_TYPE(s), the High Frequency Surname Variant Threshold (HFSV Threshold).

3.12.4.43.3. The LFP will access the Hispanic Parameter Data Store to determine the HFSV Threshold that the digraph value must pass for the LF SN to qualify as a variant of a HFSN_TYPE.

3.12.4.43.4. If the LF SN segment passes the HFSV Threshold, the LFP will assign HFSN_VAR Key(s) to the LF SN segment.

3.12.4.43.4.1. The HFSN_VAR Key will be the ID_NO associated with the HFSN_VAR that is equal to the HFSN_TYPE.

3.12.4.43.4.1.1. That is, the LF SN segment will be associated with the parent HFSN_TYPE only.

3.12.4.43.4.1.2. A LF SN may be a variant of multiple HFSN_TYPEs and may therefore receive multiple HFSN_VAR Keys.

3.12.4.43.4.2. The calculated digraph value will be associated with each HFSN_VAR Key.

3.12.4.43.5. If, by virtue of this process, all LF SN segments in a query are set equal to HFSN_VAR Keys, the LFP will direct the query record to the HFP (Section 3.11.) for generation of Given Name Keys, submission to the High Frequency Decision Matrix (HDM) and identification of retrieval criteria.

3.12.4.44. **Generating LFDIKEYs**

3.12.4.44.1. The LFDIKEY is

1) a set of digraphs formed from the LF SN segment beginning with the leftmost character and
2) a set of positional variants on those digraphs.

3.12.4.44.2. Positional information will be associated with each digraph.

3.12.4.44.3. **Base Keys**

3.12.4.44.4. The LFP will begin with the leftmost character and generate up to four digraph keys (Base Keys) from the (up to) five leftmost characters of the LF SN segment.

3.12.4.44.4.1. The first two characters form a digraph, the second and third characters form a digraph, the third and fourth characters form a digraph and the fourth and fifth characters form a digraph.

3.12.4.44.4.2. Positional information will be included: 1, 2, 3, 4, respectively: DI1, DI2, DI3, DI4.

3.12.4.44.5. If the LF SN segment has fewer than five characters, the LFP will generate fewer than four Base Keys, up to the number of characters in the LF SN.

3.12.4.44.6. Positional information will be included.

3.12.4.44.7. **Position Keys**

3.12.4.44.8. The LFP will generate from the Base Keys up to six additional Position Keys from the Base Keys.

3.12.4.44.8.1. A maximum of ten keys (Base + Position) will be generated.

3.12.4.44.8.2. The Position Keys have the same characters as the Base Keys but contain different positional information.

3.12.4.44.8.3. **For segments with 5 or more characters**:

3.12.4.44.8.4. The LFP will produce a Position Key on the first Base Key with Position 2.

3.12.4.44.8.5. The LFP will produce Position Keys on the second Base Key with Position 1 and Position 3.

3.12.4.44.8.6. The LFP will produce Position Keys on the third Base Key with Position 2 and Position 4.

3.12.4.44.8.7. The LFP will produce a Position Key on the fourth Base Key with Position 3. No Position Key is generated for Position 5 because the maximum of 10 keys has been reached.

3.12.4.44.8.8. **For segments with fewer than 5 characters**:

3.12.4.44.8.9. The LFP will produce Position Keys in the same way as for longer LF SN segments.

3.12.4.44.8.9.1. No Position Key will be generated for the final Base Key with a position to the right of the final character.

3.12.4.44.8.9.2. The total number of LFDIKEYs will be fewer than with a longer LF SN segment.

3.12.4.44.8.9.3. In GOMA, the LFP will generate a total of 7 keys: the Base Keys GO1, OM2 and MA3, and the Position Keys, GO2, OM1, OM3, and MA2. (Note: No MA4 Position Key is produced for the final digraph.)

Figure 21: Example: LFDIKEYs for LF SN Segments

LF SN SEGMENT	LFDIKEYS: BASE KEYS	LFDIKEYS: POSITION KEYS
CARRIOS	CA1 / AR2 / RR3 / RI4	CA2 / AR1 / AR3 / RR2 / RR4 / RI3
BALA	BA1 / AL2 / LA3	BA2 / AL1 / AL3 / LA2

3.12.4.44.9. **Building LF Retrieval Keys (Query)**

3.12.4.44.10. **General**

3.12.4.44.11. Each LF SN segment has been assigned a DI_KEY or set of DI_KEYs.

3.12.4.44.12. A LF SN segment may also have been assigned one or more HFSN_VAR Keys.

3.12.4.44.13. The LFP has sent queries with all HFSN_KEYs and/or HFSN_VAR Keys (including SN_INIT Keys) to the HFP for further processing.

3.12.4.44.14. The LFP will build sets of retrieval keys for mixed frequency queries (at least one HF key and one LF SN key in the string) and for queries with all low frequency keys (all SN in the string must be DI_KEYs).

3.12.4.44.14.1. A single query may have various formats – all HF, mixed and/or all-LF SN depending on the results of LF processing prior to this stage.

3.12.4.44.15. **Mixed frequency queries will not contain SN_INIT Keys.**

3.12.4.44.15.1. SN_INIT Keys may occur with HF SN keys, in which case the record will be treated as an all-HF SN record.

3.12.4.44.15.2. SN_INIT Keys may occur with LF SN keys, in which case the record will be treated as an all-LF SN record.

3.12.4.44.16. **Queries with Mixed Frequency (HF + LF) Surnames**

3.12.4.44.17. **Type 1**:

3.12.4.44.18. If one SN in the query is a HF SN and has an associated HFSN_KEY and one SN in the query record is a LF SN segment and has associated DI_KEYs, the LFP will build a Mixed Key of the HFSN_KEY and each DI_KEY (and the associated DI_VALs).

3.12.4.44.18.1. The HFSN_KEY represents a set of variants of one HFSN_TYPE.

3.12.4.44.18.1.1. GARCIA, GARCA, GARZA are all digraph variants of the HFSN_TYPE GARCIA, which has the SET_ID 0001.

3.12.4.44.18.1.2. Record adds and queries will already have been assigned the HFSN_KEY through the HFP.

3.12.4.44.18.2. The DI_KEY represents a *single* low frequency surname type that has qualified through the LF SN selection process.

Figure 22: Example: Building Mixed HF/LF SN Retrieval Keys with HFSN_KEY and DI_KEYs

QUERY NAME: GARCIA FLORENZAN	HFSN_KEY and (LF) DI_KEY
GARCIA (HF) FLORENZAN (LF)	GARCIA → 001 FLORENZAN → 000189
GARCIA (HF) LORENZ (LF)	GARCIA → 001 LORENZ → 000412

3.12.4.44.19. **Type 2**:

3.12.4.44.20. If one SN in the query record is a HF SN and has an associated HFSN_VAR Key (generated by the LFP) and one SN in the query record is a LF SN segment with associated DI_KEY(s), the LFP will build a Mixed Key of the HFSN_VAR Key and the DI_KEY for each qualifying LFSN_TYPE.

3.12.4.44.20.1. The HFSN_VAR represents a *single* HFSN_TYPE and *not* a set of variants.

3.12.4.44.20.1.1. Record adds and queries will already have been assigned the HFSN_VAR Key through the LFP.

3.12.4.44.20.2. The DI_KEY represents a ***single*** low frequency surname type that has qualified through the LF SN selection process.

Figure 23: Example: Building Mixed HF/LF SN Retrieval Keys with HFSN_VAR Keys and DI_KEYs

QUERY NAME: GARCIA FLORENZAN	HFSN_VAR and (LF) DI_KEY
BOMEZ (LF → HFSN_VAR) FLORENZAN (LF)	BOMEZ → 016978 FLORENZAN → 000189
BOMEZ (LF → HFSN_VAR) LORENZ (LF)	BOMEZ → 016978 LORENZ → 000412

3.12.4.44.21. It is likely that there will be multiple DI_KEYs for each LF SN segment, resulting in multiple Mixed Keys.

3.12.4.44.22. Once the Mixed SN Keys have been generated (and DI_VALs associated with the appropriate keys), the LFP will send any query that contains mixed HF and LF Keys to the HFP (Section 3.11.4.5) for Given Name processing and identification of retrieval criteria from the Hispanic Decision Matrix.

3.12.4.44.23. **Queries with All Low Frequency (LF + LF) Surnames**

3.12.4.44.24. The LFP will identify the LF Keys associated with query formats made up solely of LF SN segments (or a LF SN segment and SN_INIT Key(s)).

3.12.4.44.24.1. The LFP has qualified one or more LF SN segments from the LFST as variants of each LF query SN.

3.12.4.44.24.2. Each qualifying LF segment has been assigned a LF Key, the DI_KEY, and has an associated digraph value, DI_VAL.

Figure 24: Example: Low Frequency DI_KEYs and Associated Digraph Values

QUERY NAME: TOREAT FLORENZAN	LFST ID_NO	DI_KEYs + DI_VAL
TOREAT	TOREAT → 000714	000714 (1.00)
THORET	THORET → 000652	000652 (0.57)
FLORENZAN	FLORENZAN → 000189	000189 (1.00)
FLORESZ	FLORESZ → 000232	000232 (0.56)
LORENZ	LORENZ → 000412	000412 (0.59)

3.12.4.44.25. The LFP will direct a query record with all DI_KEYs and their digraph values (or DI_KEYs and SN_INIT Keys) to the Hispanic Search Engine for retrieval of database records.

3.12.5. Subordinates

None.

3.13. HISPANIC SEARCH ENGINE MODULE DECOMPOSITION

3.13.1. Identification

This module is known as the Hispanic Search Engine (HSE).

3.13.2. Type

3.13.2.1. The HSE is a function that applies to queries only.

3.13.2.2. The HSE will accept name keys and retrieval criteria from the HFP and the LFP.

3.13.2.3. The module must follow the HFP and LFP.

3.13.3. Purpose

The HSE will retrieve records from the VLDB based on criteria identified by the High Frequency Processor and the Low Frequency Processor. These criteria will delimit the set of records that can qualify for retrieval. The system must be sure that the criteria have all been identified and can be associated with database records (whether through database design and/or key generation).

3.13.4. Function

3.13.5. HNA-E will not handle records with Last Name Unknown (LNU).

3.13.6. The HSE will permit First Name Unknown (FNU).

3.13.6.1. The processing of FNU will supersede other GN restrictions.

3.13.6.2. The HSE will retrieve any database GN when FNU occurs in the query.

3.13.6.3. The HSE will retrieve any FNU in the database for any query GN.

3.13.7. High Frequency Retrieval

3.13.7.1. High frequency retrieval will include records with HFSN_KEYs, HFSN_VAR Keys and SN_INIT Keys that occur with the HF SN keys.

3.13.7.1.1. The SN_INIT Key will result in the retrieval of records that begin with or are equal to the variant initials identified by the SN_INIT Key.

3.13.7.1.2. The SN_INIT Key is stored with each SN segment.

3.13.7.1.3. All HF retrieval restrictions apply to the SN_INIT Key, as if it were a HF segment *except* that

3.13.7.1.3.1. If the SN_INIT Key is the only key in the format, the HSE will not undertake a database search.

3.13.7.1.4. No further, separate detailing of the SN_INIT Key is given.

3.13.7.2. High frequency retrieval will include mixed HF and LF SN Keys but with no SN_INIT Keys.

3.13.8. **All HFSN_KEYs (or SN_INIT Key)**

3.13.9. For queries with all HFSN_KEYs, the HSE will retrieve records from the database records that

- contain the appropriate SN format (position, more/fewer segments, different segments) as specified in the HDM,
- contain the appropriate HFSN_KEYs,
- meet all the criteria identified in the HDM and
- meet the GN restrictions.

3.13.10. The HFSN_KEY will result in retrieval of the HFSN_TYPE and all its variants.

3.13.10.1. The HSE will further restrict the retrieval to records that match at least one key of the GN.

3.13.10.1.1. If the query has produced only HFGN_KEYs, only records that have at least one of the HFGN_KEYs will be retrieved.

3.13.10.1.2. If the query has produced mixed HFGN_KEYs and GN_INIT Keys, the HSE will retrieve records that match at least one of the HFGN_KEY or one of the GN_INIT Keys.

3.13.10.1.3. If the query has produced only GN_INIT Keys, the HSE will retrieve records that match at least one of the GN_INIT Keys.

Figure 25: Example: Record Matching Criteria: All HFSN_KEYs and HFGN_KEYs

QUERY #1	**RODRIGUEZ**	**LOPEZ**	**JOSE**	**CARLOS**	**CRITERIA**
HFSN_KEY	002	010			
HFGN_KEY			0001	0007	
HDM FORMATS:					
1	RODRIGUEZ (002)	LOPEZ (010)			YOB5, RL4, MFU, GN contains 0001 or 0007
2	LOPEZ	RODRIGUEZ			YOB4, RL4, MFU, GN contains 0001 or 0007
3	RODRIGUEZ				YOB4, RL4, MFU, GN contains 0001 or 0007
4	LOPEZ				YOB2, RL1, MFU, GN contains 0001 or 0007
5	RODRIGUEZ	* (ANY SN)			YOB2, RL1, FU, GN contains 0001 or 0007
6	LOPEZ	* (ANY SN)			YOB0, RL0, MFU, GN contains 0001 or 0007
7	* (ANY SN)	RODRIGUEZ			YOB0, RL0, MFU, GN contains 0001 or 0007
8	* (ANY SN)	LOPEZ			YOB0, RL0, MFU, GN contains 0001 or 0007

Figure 26: Example: Record Matching Criteria: All HFSN_KEYs and Mixed HFGN_KEYs and GN_INIT Keys

QUERY #2	RODRIGUEZ	LOPEZ	JOSSE	CARLOS	CRITERIA
HFSN_KEY	002	010			
HFGN_KEY				0007	
GN_INIT Key(s)			041 (J, H)		
HDM FORMATS:					
1	RODRIGUEZ (002)	LOPEZ (010)			YOB5, RL4, MFU, GN initial = J or H; or GN= 0007
2	LOPEZ	RODRIGUEZ			YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
3	RODRIGUEZ				YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
4	LOPEZ				YOB2, RL1, MFU, GN initial = J or H; or GN= 0007
5	RODRIGUEZ	*			YOB2, RL1, FU, GN initial = J or H; or GN= 0007
6	LOPEZ	*			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
7	*	RODRIGUEZ			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
8	*	LOPEZ			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007

3.13.11. **HFSN_KEY and/or HFSN_VAR Keys (or SN_INIT Key)**

3.13.12. For queries with mixed HFSN_KEYs and HFSN_VAR Keys and queries with all HFSN_VAR Keys, the HSE will retrieve records from the database records that

- contain the appropriate HFSN_KEYs and/or HFSN_VAR Keys,

- contain the appropriate SN format (position, more/fewer segments, different segments) as specified in the HDM,
- meet all the criteria identified in the HDM and
- meet the GN restrictions.

3.13.13. The HFSN_VAR Key will retrieve a ***single*** HFSN_TYPE and not the set of variants associated with the HFSN_TYPE (e.g., the name LOPEZ but not all its variants; the variants will be retrieved by the LFP).

3.13.13.1. The HSE will further restrict the retrieval to records that match at least one key of the GN.

3.13.13.1.1. If the query has produced only HFGN_KEYs, only records that have at least one of the HFGN_KEYs will be retrieved.

3.13.13.1.2. If the query has produced mixed HFGN_KEYs and GN_INIT Keys, the HSE will retrieve records that match at least one of the HFGN_KEY or the GN_INIT Keys.

3.13.13.1.3. If the query has produced only GN_INIT Keys, the HSE will retrieve records that match at least one of the GN_INIT Keys.

Figure 27: Example: Record Matching Criteria: All HFSN_KEY and/or HFSN_VAR Keys

QUERY #1	RODRIGUEZ	SLOPEZ	JOSSE	CARLOS	CRITERIA
HFSN_KEY	002				
HFSN_VAR Key		00976			
HFGN_KEY				0007	
GN_INIT Key(s)			041 (J, H)		
HDM FORMATS:					
1	RODRIGUEZ (002)	LOPEZ (000976)			YOB5, RL4, MFU, GN initial = J or H; or GN= 0007
2	LOPEZ	RODRIGUEZ			YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
3	RODRIGUEZ				YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
4	LOPEZ				YOB2, RL1, MFU, GN initial = J or H; or GN= 0007
5	RODRIGUEZ	*			YOB2, RL1, FU, GN initial = J or H; or GN= 0007
6	LOPEZ	*			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
7	*	RODRIGUEZ			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
8	*	LOPEZ			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007

3.13.14. **Mixed HFSN_KEY and/or HFSN_VAR Keys and LF DI_KEYs (*no* SN_INIT Key)**

3.13.15. For queries with mixed HFSN_KEYs/HFSN_VAR Keys and LF DI_KEYs, the HSE will retrieve records from the database records that

- contain the appropriate HFSN_KEYs/HFSN_VAR Keys and DI_KEYs,
- contain the appropriate SN format (position, more/fewer segments, different segments) as retrieved from the HDM,
- meet all the criteria identified in the HDM and
- meet the GN restrictions.

3.13.16. The LFP generated (multiple) query formats that contain a HF Key and a DI_KEY.

3.13.16.1. The DI_KEY will retrieve an exact match on a ***single*** LFSN_TYPE.

3.13.16.2. Each HFSN_KEY or HFSN_VAR Key may participate in query formats with several different DI_KEYs that were identified as variants by the LFP.

3.13.16.3. Each query format will serve as a different query.

3.13.17. The HSE will further restrict the retrieval to records that match at least one key of the GN.

3.13.17.1. If the query has produced only HFGN_KEYs, only records that have at least one of the HFGN_KEYs will be retrieved.

3.13.17.2. If the query has produced mixed HFGN_KEYs and GN_INIT Keys, the HSE will retrieve records that match at least one of the HFGN_KEY or the GN_INIT Keys.

3.13.17.3. If the query has produced only GN_INIT Keys, the HSE will retrieve records that match at least one of the GN_INIT Keys.

Figure 28: Example: Record Matching Criteria: Mixed HFSN_KEYs/HFSN_VAR Keys and LF DI_KEYs

QUERY #1	THORET	SLOPEZ	JOSSE	CARLOS	CRITERIA
HFSN_KEY					
HFSN_VAR Key		00976			
DI_KEY	000652 (THORET)				
	000714 (TOREAT)				
HFGN_KEY				0007	
GN_INIT Key(s)			041 (J, H)		
HDM FORMATS:					
1	THORET	LOPEZ (000976)			YOB5, RL4, MFU, GN initial = J or H; or GN= 0007
2	LOPEZ	THORET			YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
3	THORET				YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
4	LOPEZ				YOB2, RL1, MFU, GN initial = J or H; or GN= 0007
5	THORET	*			YOB2, RL1, FU, GN initial = J or H; or GN= 0007
6	LOPEZ	*			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
7	*	THORET			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
8	*	LOPEZ			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
1	TOREAT	LOPEZ			YOB5, RL4, MFU, GN initial = J or H; or GN= 0007
2	LOPEZ	TOREAT			YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
3	TOREAT				YOB4, RL4, MFU, GN initial = J or H; or GN= 0007
4	LOPEZ				YOB2, RL1, MFU, GN initial = J or H; or GN= 0007
5	TOREAT	*			YOB2, RL1, FU, GN initial = J or H; or GN= 0007
6	LOPEZ	*			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
7	*	TOREAT			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007
8	*	LOPEZ			YOB0, RL0, MFU, GN initial = J or H; or GN= 0007

3.13.18. The HSE will not retrieve database records that have already been retrieved with another key.

3.13.19. The HSE will retrieve the database record ID; the Dual-SN Formats; keys, their segment position and their related DI_VALs; Record Gender; and TAQ tags.

3.13.20. **Low Frequency Retrieval**

3.13.21. The HSE will retrieve records from the database that contain one or both of the query DI_KEYs in any SN position in the database record.

3.13.21.1. The HSE will retrieve records that contain the DI_KEYs within a specified YOB Range for a Refusal Code Level.

3.13.21.2. The HSE will access the RLYOB Data Store to determine the Refusal Code Level and associated Year-of-Birth Range that will apply.

3.13.21.3. The HSE will retrieve all records from the database with

- both DI_KEYs (or one DI_KEY and one SN_INIT Key) in either position and RLYOB restriction;
- one of the DI_KEYs alone and RLYOB restriction; and
- one DI_KEY in either position, if the Year-of-Birth Range is YOB2 and the Refusal Code Level is RL1 (i.e., 00 or Type 1 Serious).

3.13.22. The HSE will retrieve the database record; the record ID; the Dual-SN Formats; keys, their segment position and their related DI_VALs; Record Gender; and TAQ tags.

3.13.23. The HSE will not retrieve a record that has already been retrieved using other access methods (i.e., Mixed Frequency SN or HF names).

3.13.24. All records retrieved from the database will be sent to the Hispanic Filter and Sorter.

3.14. HISPANIC FILTER AND SORTER MODULE DECOMPOSITION

3.14.1. Identification

This module is known as the Hispanic Filter and Sorter (HFS).

3.14.2. **Type**

3.14.2.1. The HFS is a module that accepts database records retrieved by the HSE.

3.14.2.2. The HFS compares each database record to the query record to determine if it qualifies for return to the user.

3.14.2.3. The HFS is constituted of two subordinate functions: the Hispanic Filter and the Hispanic Sorter.

3.14.2.4. The HFS must follow the Hispanic Search Engine (HSE).

3.14.3. **Purpose**

3.14.3.1. The set of database records that the HSE will retrieve will be a set of records delimited by quite narrow retrieval criteria. The database records will have a digraph value associated with most SN segments and with many GN segments. However, the relative value of the database records to the query record will not be clear. The HFS will, therefore, evaluate each of the records retrieved for its proximity to a query record, will retain those that pass a pre-established threshold and will sort the resultant candidate list.

3.14.3.2. The filtering process will take into account a number of factors that play a role in determining the relative value of Hispanic *names*.

3.14.3.3. The filtering process will take into account factors that aid in the determination of the relative value of a Hispanic *records*.

3.14.4. **Function**

The HFS will first compare and qualify the query name and database record name to determine a surname value (SN_VAL), will then evaluate and qualify the query name and database record to determine a given name value (GN_VAL) and will generate a composite score for the database records that qualified on the basis of name evaluation by factoring in values for Date-of-Birth, Refusal Level and Country of Birth.

The first comparison will be to identify an exact record match. All other comparison will be between the Dual-SN Format of the query and database record (for records with more than two surnames).

3.14.4.1. **Filter Function of the HFS**

3.14.4.2. **General**

3.14.4.3. The Hispanic Filter and Sorter (HFS) will accept the candidate database records retrieved by the HSE.

3.14.4.4. The HFS will first determine if the query record and database record match exactly.

3.14.4.4.1. The HFS will compare the base format of the query and database record; i.e., no derived format.

3.14.4.4.2. The name (both SN and GN), Date-of-Birth and Country-of-Birth must match exactly.

3.14.4.4.3. If the query and database records match exactly, the HFS will tag the record as an exact match and send the record directly to the Sorter Function of the HFS.

3.14.4.5. The HFS will calculate name scores for each candidate database record as it compares to the query record.

3.14.4.5.1. The HFS will use the derived formats as the basis of record comparison.

3.14.4.5.2. A score for the SN, the SN_VAL, will be calculated.

3.14.4.5.3. A score for the GN, the GN_VAL, will be calculated.

3.14.4.5.3.1. The HFS will adjust the digraph value retrieved with the database record by multiplying that value by factors assigned to several parameters.

3.14.4.5.3.2. Factors (see Section 4.13) that contribute to the determination and evaluation of the name score (SN_VAL and GN_VAL) include

- SNTHR
- GNTHR
- ASVAL
- AGVAL
- OPSVAL
- OPGVAL
- INITSN
- INITGN
- TAQASN
- TAQAGN
- TAQXSN
- TAQXGN
- RGNDR

3.14.4.5.3.3. To be included in the final candidate list, the score of the SN and the score of the GN must each pass pre-determined SN and GN threshold levels (SNTHR and GNTHR).

3.14.4.6. **Surname Evaluation**

3.14.4.7. A candidate record must pass a SN evaluation before it will be submitted to a GN evaluation.

3.14.4.8. No record with Last Name Unknown (LNU) will be handled by HNA-E.

3.14.4.9. The SN evaluation will be performed on the dervied formats (including the Dual-SN Formats) associated with the query and database records.

3.14.4.10. **High Frequency SN Keys (HFSN_KEYs or HFSN_VAR Keys)**

3.14.4.10.1. The HFS will compare the keys of the query and database and assign the DI_VAL retrieved with the database record to the SN Comparands with matching keys.

3.14.4.10.1.1. Only one assignment of DI_VAL can be made for a match.

3.14.4.10.1.2. If the query is GARCIA GOMEZ and the database record is GARCIA GARCIA, the HFS will assign the DI_VAL to one GARCIA match only.

3.14.4.10.2. If the SN Keys do not match, the HFS will perform a digraph match of the segments with no assigned value (LOPEZ and GOMEZ in Figure 29) and will assign the digraph score to the DI_VAL.

Figure 29: Example: Database Records with HFSN_KEYs to be Evaluated by HFS

	SN#1	HFSN_KEY	DI_VAL	SN#2	HFSN_KEY	DI_VAL
QUERY	GARCIA	0001		GOMEZ	0010	
DATABASE RECORDS	GARCIA	0001	1.00	BOMEZ	0010	0.67
	BARCIA	0001	0.71	GAMEZ	0010	0.67
	LOPEZ	0004	0.17	GARCIA	0001	1.00

3.14.4.11. **Low Frequency SN Keys** (DI_KEYs)

3.14.4.11.1. The HFS will assign the DI_VAL associated with the DI_KEY to matching database and query DI_KEYs.

3.14.4.11.1.1. Only one assignment of DI_VAL can be made for a match.

3.14.4.11.1.2. If the query is THORET FLORENZAN and the database record is THORET THORET, the HFS will assign the DI_VAL to one THORET match only.

3.14.4.12. If the SN Keys do not match, the HFS will perform a digraph match of the segments with no assigned value (LOPEZ and GARCIA in Figure 30) and will assign the digraph score to the DI_VAL.

3.14.4.12.1. If there is more than one pair that does not have an assigned digraph value, the HFS will perform a digraph evaluation for each of the pairs. (See Section 3.14.4.16 for details of the digraph assignment.)

3.14.4.12.2. Each value will be submitted to parameter evaluation.

3.14.4.12.3. After all parameters have been applied, the HFS will choose the highest score for each pair. (See Section 3.14.4.17)

Figure 30: Example: Database Records with LF SN (Mixed or all LF Keys) to be Evaluated by HFS

	SN#1	HFSN_KEY/ DI-KEY	DI_VAL	SN#2	HFSN_KEY/ DI_KEY	DI_VAL
QUERY	GARCIA	0001		THORET	000652	
DATABASE RECORDS	GARCIA	0001	1.00	THORET	000652	1.00
	THORET	000652	1.00	BARCIA	0001	0.71
	LOPEZ	0004	0.00	THORET	000652	1.00

3.14.4.13. The HFS will adjust the DI_VAL of each segment according to parameter values in the Hispanic Parameter Data Store (see Section 4.13 for details).

3.14.4.13.1. The HFS will determine if the appropriate parameter conditions obtain.

3.14.4.13.2. If the appropriate conditions are present, the DI_VAL will be multiplied by the value assigned to the parameter and the DI_VAL will be lowered.

3.14.4.13.3. **Parameter Conditions**

3.14.4.13.4. **INITSN**: Initial

3.14.4.13.4.1. Definition 1: The SN segment is a single character in both comparands and the character matches exactly.

3.14.4.13.4.2. Action: The HFS will make no change.

3.14.4.13.4.3. Definition 2: A SN segment is a single character and its SN_INIT Key matches the SN_INIT Key of the other comparand.

3.14.4.13.4.4. Action: Assign the INITSN value to the comparison value (i.e., do not calculate the DI_VAL). The initial may be subjected to any following actions (e.g., out-of-place segment).

3.14.4.13.4.5. Definition 3: A SN segment is a single character and the SN_INIT Keys of the comparands do not match.

3.14.4.13.4.6. Action: Assign the INITNM value to the comparison value (i.e., do not calculate the DI_VAL). The initial may be subjected to any following actions (e.g., out-of-place segment).

3.14.4.13.5. **OPSVAL**: Out-of-Place Surname Value

3.14.4.13.5.1. Definition: A SN segment that is not in the same relative position in the SN string in both the database and query records.

3.14.4.13.5.2. Action: Multiply the DI_VAL by the OPSVAL.

3.14.4.13.6. **ASVAL**: Anchor Surname Value

3.14.4.13.6.1. Definition: For database records that contain two SN segments, the database SN segments are in the correct position relative to the query SN segments.

3.14.4.13.6.2. Action: Multiply the DI_VAL of the second (rightmost) segment by the ASVAL.

Figure 31: Example 1: SN Parameter Evaluation: OPSN Applies

	GARCIA	**GOMEZ**
BOMEZ		0.67 * 0.65 = 0.44
GARCIA	1.00 * 0.65 = 0.65	

Figure 32: Example 2: SN Parameter Evaluation: OPSN Applies

	GARCIA	**GOMEZ**
GAMEZ		0.67 * 0.65 = 0.44

Figure 33: Example 3: SN Parameter Evaluation: ASVAL Applies

	GARCIA	**GOMEZ**
GARZA	0.62	
GOMEZ		1.00 * 0.65 = 0.65

3.14.4.13.7. **TAQ Filter**

3.14.4.13.8. All TAQ tags (ID_NO, disposition, TAQ_TYPE and associated SN stem) will be retrieved with the database record.

3.14.4.13.9. The HFS will evaluate any TAQs associated with the SN segments being evaluated, except Stranded Prefixes (see Section 3.5.4.2.7.3).

3.14.4.13.9.1. A Stranded Prefix will not play a role in the record comparison.

3.14.4.13.10. **Single TAQs**

3.14.4.13.11. **Missing TAQs**

3.14.4.13.12. TAQASN: Absent TAQ Value

3.14.4.13.12.1. Definition 1: One of the two comparands (query/database SN segment) has a TAQ tag, the other does not.

3.14.4.13.12.2. Definition 2: Both comparands (query/database SN segments) have a single TAQ tag, one is a TAQ DELETE, the other a TAQ DISREGARD.

3.14.4.13.12.3. Action: Multiply the DI_VAL by the TAQASN value.

Figure 34: Example: TAQ DISREGARD (DE) and No TAQ

	DE VARGAS
VARGAS	1.00 * 0.90 = 0.90

Figure 35: Example: TAQ DISREGARD (DE) and TAQ DELETE (DR)

	DE VARGAS
DR VARGAS	1.00 * 0.90 = 0.90

3.14.4.13.13. **TAQ DELETE**

3.14.4.13.14. TAQXSN: Deleted TAQ Value

3.14.4.13.14.1. Definition: Both SN comparands have a single TAQ DELETE tag.

3.14.4.13.14.2. Action:

3.14.4.13.14.3. If the TAQ DELETE tags refer to the same TAQ segment, the DI_VAL will be unchanged.

3.14.4.13.14.4. If the TAQ DELETE tags refer to different TAQ DELETE segments, multiply the DI_VAL by the TAQXSN value.

Figure 36: Example: Same TAQ DELETE (DR)

	DR VARGAS
DR VARGAS	1.00

Figure 37: Example: Different TAQ DELETEs (DR and SR)

	SR VARGAS
DR VARGAS	1.00 * 0.850 = 0.85

3.14.4.13.15. **TAQ DISREGARD**

3.14.4.13.15.1. Definition: The HFS will access the TAQ Filter Data Store (TF) to process records that both contain SN TAQ segments that have been tagged as DISREGARD.

3.14.4.13.15.2. Action 1: The HFS will assign TAQDIS#1 to the TAQ DISREGARD segment for the database SN segment and TAQDIS#2 to the TAQ DISREGARD segment for the query SN segment.

3.14.4.13.15.3. Action 2: If the two TAQ DISREGARD segments match, the DI_VAL will remain unchanged.

3.14.4.13.15.4. Action 3: If the two TAQ DISREGARD segments do not match, the HFS will identify the TF_VALUE for the pair in the TF.

3.14.4.13.15.4.1. The HFS will multiply the DI_VAL by the TF_VALUE for the pair.

Figure 38: Example: Different TAQ DISREGARDs (DE and LA)

	DE PENA
LA PENA	1.00 * 0.75 = 0.75

3.14.4.13.16. **Multipart TAQs**

3.14.4.13.16.1. Definition: If at least one SN comparand has multipart TAQ tags (they may be all DISREGARD, all DELETE, or mixed DISREGARD/DELETE), the HFS will perform the following analyses.

3.14.4.13.16.2. Action: If all TAQs match, HFS will make no change in the DI_VAL.

3.14.4.13.16.3. **TAQ DELETEs**

3.14.4.13.16.3.1. Definition: All DELETE tags

3.14.4.13.16.3.2. Action 1: If any DELETE TAQ matches, the HFS applies no change.

3.14.4.13.16.3.3. Action 2: If no DELETE TAQs match, multiply the DI_VAL by the TAQXSN Value.

Figure 39: Example: Multiple TAQ DELETEs with Some Match

	REV DR VARGAS
REV VARGAS	1.00

Figure 40: Example: Multiple TAQ DELETEs with No Match

	GENERAL DR VARGAS
REV SR VARGAS	1.00 * 0.85 = 0.85

3.14.4.13.16.4. **TAQ DISREGARDs**

3.14.4.13.16.4.1. Definition: All DISREGARD tags

3.14.4.13.16.4.2. Action 1: If any TAQ DISREGARD segment matches, the HFS will make no change in the DI_VAL.

3.14.4.13.16.4.3. Action 2: If no TAQ DISREGARD segments match, the HFS will identify the highest match value from the TF (TF_VALUE) and multiply that by the DI_VAL.

Figure 41: Example: Multiple TAQ DISREGARDs with Matching TAQ Segment (DE LAS/DE LOS)

	DE LAS LUNAS
DE LOS LUNAS	1.00

Figure 42: Example: Multiple TAQ DISREGARDs with No Matching TAQ Segment (DE SANTA/LA)

	DE SANTA MARIA
LA MARIA	1.00 * 0.75 = 0.75

3.14.4.13.16.5. **TAQ DISREGARD and DELETEs**

3.14.4.13.16.5.1. Definition: Mixed DISREGARD/DELETE tags

3.14.4.13.16.5.2. Action 1: If DISREGARD segments are present in both comparands and there is any match among the DISREGARD segments, the HFS will make no change in the DI_VAL.

3.14.4.13.16.5.3. Action 2: If DISREGARD segments are present in both comparands and there is no match among the DISREGARD segments, the HFS will determine the highest match value from the TF for any DISREGARD tags and multiply the DI_VAL by that value. (That is, ignore any DELETE tags.)

3.14.4.13.16.5.4. Action 3: If a DISREGARD segment is in one comparand and not the other and the two comparands have at least one DELETE tag that matches, the HFS will make no change in the DI_VAL.

3.14.4.13.16.5.5. Action 4: If a DISREGARD segment is in one comparand and not the other and the two comparands have DELETE tags that do not match, multiply the DI_VAL by the TAQXSN.

Figure 43: Example: Multiple TAQs, DISREGARDs (DE/LOS)

	SR DE VARGAS
DR LOS VARGAS	1.00* 0.75 = 0.75

Figure 44: Example: Multiple TAQs, DELETEs (DRA/DR)

	DRA DE VARGAS
DR VARGAS	1.00* 0.85 = 0.85

3.14.4.14. After all parameters have been applied, the HFS will calculate the SN_VAL.

3.14.4.14.1. The HFS will choose the highest value for the row and column for any SN segments that have more than one digraph value assigned to them.

3.14.4.14.2. The HFS will sum the DI_VALs of all SN segments and will divide by the number of DI_VALs.

Figure 45: Example 1: Filter Evaluation

	GARCIA	GOMEZ
BOMEZ		0.67 * 0.65 = 0.44
GARCIA	1.00 * 0.65 = 0.65	

Figure 46: Example 2: Filter Evaluation

	GARCIA	GOMEZ
GAMEZ		0.67 * 0.65 = 0.44

Figure 47: Example 3: Filter Evaluation

	GARCIA	GOMEZ
GARZA	0.62	
GOMEZ		1.00 * 0.65 = 0.65

3.14.4.14.3. In Figure 45, 0.44 + 0.65 / 2 = 0.55

3.14.4.14.4. In Figure 46, 0.44 / 1 = 0.44

3.14.4.14.5. In Figure 47, 0.62 + 0.65 / 2 = 0.64

3.14.4.15. The HFS will compare the SN_VAL to the SNTHR.

3.14.4.15.1. The SN_VAL must be equal to or greater than the SNTHR.

3.14.4.15.2. If the SNTHR were 0.60, only Example 3 above would pass.

3.14.4.15.3. The record must pass the SNTHR to qualify for Given Name Evaluation.

3.14.4.16. **Given Name Evaluation**

3.14.4.16.1. The HFS will evaluate the GN in a similar way to the SN evaluation.

3.14.4.16.2. The HFS will assign a DI_VAL of 1.00 to any match with FNU.

3.14.4.16.3. The GN format will permit more than two GN segments to be evaluated.

3.14.4.16.3.1. If the segment pair has a matching HFGN_KEY, the digraph value (DI_VAL) retrieved with that key will be assigned to the pair being evaluated.

3.14.4.16.3.2. For any segment pair that does not have a HFGN_KEY and associated DI_VAL, the DI_VAL will be calculated. (See Section 3.11.4.19.4 for digraph evaluation.)

3.14.4.16.3.3. The HFS will not calculate a digraph value for a GN_INIT Key value or GN initial.

3.14.4.16.3.3.1. The HFS will calculate the digraph relationship for all segments that have not been assigned a DI_VAL.

3.14.4.16.3.3.2. The HFS will not compare names that have a DI_VAL assigned.

Figure 48: Example: GN Digraph Evaluation

	MARIA	**LORNA**	**SILVIA**	**CATERINA**
CATHERINA				0.74 (HFGN_KEY)
MARIA	1.00 (HFGN_KEY)			
LARA		0.36	0.08	

MILDRED		0.00	0.07	

3.14.4.16.4. The DI_VAL of each GN segment will be adjusted by several GN parameters.

3.14.4.16.5. **INITGN**: Given Name Initial

3.14.4.16.5.1. Definition 1: The GN segment is a single character in both comparands and the character matches exactly.

3.14.4.16.5.2. Action: The HFS will make no change.

3.14.4.16.5.3. Definition 2: A GN segment is a single character and its GN_INIT Key matches the GN_INIT Key of the other comparand.

3.14.4.16.5.4. Action: Assign the INITGN value to the comparison value (i.e., do not calculate the DI_VAL). The initial may be subjected to any following actions (e.g., out-of-place segment).

3.14.4.16.5.5. Definition 3: A GN segment is a single character and the GN_INIT Keys of the comparands do not match.

3.14.4.16.5.6. Action: Assign the INITNM value to the comparison value (i.e., do not calculate the DI_VAL). The initial may be subjected to any following actions (e.g., out-of-place segment).

3.14.4.16.6. **OPGVAL**: Out-of-Place Given Name Value

3.14.4.16.6.1. Definition: A GN segment that is not in the same relative position in the GN string in both the database and query records.

3.14.4.16.6.2. Action: Multiply the DI_VAL by the OPGVAL.

3.14.4.16.7. **AGVAL**: Anchor Given Name Value

3.14.4.16.7.1. Definition: For database records that contain two or more GN segments, the database SN segments are in the correct position relative to the query SN segments.

3.14.4.16.7.2. Action: Multiply the DI_VAL of the GN segments to the right of the first (leftmost segment) by the AGVAL.

Figure 49: Example 1: GN Parameter Evaluation: OPGN Applies

	MARIA	**CATHERINA**
KATHERINA		0.90 * 0.65 = 0.59
MARIA	1.00 * 0.65 = 0.65	

Figure 50: Example 2: GN Parameter Evaluation: OPGN Applies

	JOSE	BARTOLOMEO
BARTO		0.71 * 0.65 = 0.46

Figure 51: Example 3: GN Parameter Evaluation: AGVAL Applies

	JUAN	MARIO
JUANA	0.73	
MARIA		0.83 * 0.65 = 0.54

Figure 52: Example 4: Given Name Parameter Evaluation

	MARIA	LARA	MILDRED	CATERINA
CATHERINA				0.74*0.65 = 0.48 (OPGVAL)
MARIA	1.00*0.65 = 0.65 (OPGVAL)			
LORNA		0.36*0.65 = 0.23 (OPGVAL)	0.08*0.65 = 0.05 (OPGVAL)	
SILVIA		0.00*0.65 = 0.00 (OPGVAL)	0.07*0.65 = 0.05 (OPGVAL)	

3.14.4.16.8. TAQ Evaluation will proceed as with the SN, mutatis mutandi (See Section 3.14.4.13.7).

3.14.4.17. After all GN evaluations have been performed, the HFS will choose the highest score for each GN segment that has multiple DI_VALs (i.e., those for which no DI_VAL was retrieved with the key).

3.14.4.17.1. The highest score for **both** the row and column must be chosen.

3.14.4.17.2. Only one score per row and column is permitted.

3.14.4.17.3. If two scores are equal, only one is chosen.

3.14.4.17.4. In the example above, the higher score for LORNA is on the match with LARA (0.23); for SILVIA, MILDRED (0.05).

3.14.4.17.4.1. Note that MILDRED scores are equal, but the row for LORNA has already been chosen.

3.14.4.17.4.2. Only one value can be chosen for each row and column.

3.14.4.18. The HFS will sum all DI_VALs from the comparison matrix and will divide by the number of DI_VALs to produce the GN score.

3.14.4.18.1. In Example 1, 0.59 + 0.65/2 = 0.62

3.14.4.18.2. In Example 2, 0.46/1 = 0.46

3.14.4.18.3. In Example 3, 0.73 + 0.54/2 = 0.64

3.14.4.18.4. In Example 4, 0.48 + 0.65 + 0.23 + 0.05/4 = 0.33

3.14.4.19. The HFS will further evaluate the Given Name by comparing the record gender of the two comparands.

3.14.4.19.1. If the record genders match, no action will take place.

3.14.4.19.2. If the record genders do not match, the HFS will apply the RGNDR value to the GN score.

3.14.4.19.2.1. The HFS will access the TF to determine the RGNDR Value.

3.14.4.19.2.2. The HFS will multiply the GN score by the RGNDR value.

3.14.4.20. The value resulting from the full GN evaluation will be the GN_VAL.

3.14.4.21. The HFS will compare the GN_VAL to the GNTHR.

3.14.4.21.1. The GN_VAL must be equal to or greater than the GNTHR.

3.14.4.21.2. The GN_VAL must pass the GNTHR for the record to qualify for calculation of the Composite Score.

3.14.4.22. **Composite Score**

3.14.4.23. The HFS will develop a composite score for two comparands that will reflect the proximity of the query and database *records*.

3.14.4.24. The Composite Score will be used to rank order the records being evaluated.

3.14.4.25. The Name is one component of the Composite Score; others are the Refusal Level, Date-of-Birth and Country of Birth.

3.14.4.25.1. The HFS will adjust the GN_VAL and the SN_VAL by factors that reflect the proximity of the Refusal Level, Date-of-Birth and Country of Birth.

3.14.4.25.2. The GN_VAL and SN_VAL will be multiplied by RL, YOB and COB factors.

3.14.4.26. **Refusal Level Factor**

3.14.4.27. The HFS will access the Refusal Code Level Data Store to determine the Refusal Level Category of the Refusal Code.

3.14.4.28. The HFS will access the Hispanic Parameter Data Store to find the PARM_VAL associated with the Refusal Level (RL#).

3.14.4.29. **Date-of-Birth Factor**

3.14.4.30. The HFS will access the Year-of-Birth Range Data Store to determine the YOB Category, YOB#, of the Dates-of-Birth of the comparands. The highest value is applied to the relationship.

3.14.4.31. The HFS will access the Hispanic Parameter Data Store to find the PARM_VAL associated with the YOB Category (YOB#).

3.14.4.32. **Country-of-Birth Factor**

3.14.4.33. The HFS will access the Hispanic Country-of-Birth Category Data Store (HCOB) to determine the COB Category, COB#.

3.14.4.33.1. The HFS will identify the COB#.

3.14.4.33.2. The HFS will access the Hispanic Parameter Data Store to find the PARM_VAL associated with the Country-of-Birth Category (COB#).

3.14.4.34. **Calculating the Composite Score**

3.14.4.35. The HFS will calculate a Composite Score by multiplying the SN_VAL by the GN_VAL by the RL# PARM_VAL by the YOB# PARM_VAL by the COB# PARM_VAL.

3.14.4.36. **Final Sort Function of the HFS**

3.14.4.37. The HFS will order the final candidate list.

3.14.4.38. The HFS will place at the top of the candidate list all records that have been tagged as exact matches.

3.14.4.39. The HFS will then rank order in descending order of Composite Score all records for which a Composite Score has been calculated.

3.14.4.39.1. The goal of the final sort is to place exact record matches on the top and to rank order the remaining records by the degree of contribution that each data element (SN, GN, DOB, COB, Refusal Code Level (RL)) makes to the overall record value.

3.14.4.39.2. Further details of the sort will be derived from extensive discussion about the business requirements.

3.14.4.39.3. Because the scores from the various pipes may not have been calculated in the same way, a method for evaluating the relative value of candidate records will have to be devised.

3.14.4.40. **Internal Sort Order**

3.14.4.40.1. There may be cases in which the sorting criteria are met equally by more than 1 record.

3.14.4.40.2. Where multiple records qualify equally, there will be an internal sort order.

3.14.4.40.2.1. SN Score

3.14.4.40.2.2. GN Score

3.14.4.40.2.3. DOB Level

3.14.4.40.2.4. Refusal Code Level

3.14.4.40.2.5. COB Relationship

3.14.4.41. The HFS will return the top n records to the central CLASS-E sorter.

3.14.4.41.1. The number of records to be returned will be a system setting.

3.15. LINGUISTIC TRACE FACILITY MODULE DECOMPOSITION

3.15.1. Identification

This module is known as the Linguistic Trace Facility (LTF).

3.15.2. Type

The LTF is a program that will interact with any or all modules and functions within those modules.

3.15.3. Purpose

The LTF will allow system evaluators to access information about the system functions so that the quality of the content can be ensured. To diagnose and remedy problems associated with questionable system results, evaluators must have access to the results of system functionality at various points during the processing cycle.

3.15.4. Function

3.15.4.1. The LTF will be a mechanism that will copy and divert statistics, information, processing results to a file outside the main processing module.

3.15.4.2. The file will be readily accessible for on-line examination by system evaluators.

3.15.4.3. Multiple trace points will be identified when the system is built.

3.15.4.4. Examples of trace points:

- Derived record formats
- All keys generated for a query and for an add

- Records qualifying with the LFDI_KEY
- SN and GN DI_VAL
- SN_VAL and GN_VAL
- Record Gender
- RL#, YOB#, COB# Values
- Sort considerations

4. DATA DECOMPOSITION

4.1. DATA

4.1.1. The input data for an HNA-E query will contain all information that is currently required by CLASS.

4.1.2. The input data for an HNA-E query will be in the standard format currently required by CLASS.

- NAME (Surname, Given Name);
  - The SN is a required name field and therefore must be filled.
  - Last Name Unknown (LNU) is not a permitted string in HNA-E.
  - The SN may be represented by a single character, which will be interpreted as an initial.
- DOB (Date of Birth; Day Month Year); and
- COB (Country of Birth; FIPS codes).

4.1.3. In addition, the following will be specified:

- Applicant Gender (AG): Male (M), Female (F), Unknown/Ambiguous (U).
- A unique identifier (UID) (as defined in CLASS-E).

4.1.4. For record adds, additional record information will be entered, as required by CLASS and CLASS-E: e.g., refusal code, province of birth.

4.2. DATA COLLECTION

4.2.1. Two alternative approaches to tagging the name data are available: the name as an object and the name as a data element.

4.2.2. The system could define the name as an object that knows something about itself and collects information as it passes through the various processing modules.

4.2.2.1. A name object would make the relevant information available to the various processing modules, as needed, from one consistent, predefined object.

4.2.2.2. A name object may also permit the same name to be handled in the same way on another occasion. Reuse of information would be especially valuable for HF names.

4.2.3. The second, alternative approach is to tag the specific items as they undergo processing or change, to access information in data stores as it is needed, and to tag the name or name segment for the relevant processes it undergoes.

4.3. DATA STORES

4.3.1. Several of the Data Stores proposed could be collapsed into one data store (e.g., the HF SN Data Stores: HFST and HFSV); for ease and clarity of exposition and reference, the data stores have been maintained separately.

4.3.2. HNA-E will access X Data Stores:

- Hispanic TAQ Data Store (HTD)
- High Frequency Surname Type Data Store (HFST)
- High Frequency Surname Variant Data Store (HFSV)
- Low Frequency Surname Type Data Store (LFST)
- Hispanic Given Name Type Data Store (HGT)
- High Frequency Given Name Variant Data Store (HFGV)
- Hispanic Character Data Store (HCD)
- Hispanic Parameter Data Store (HPD)
- High Frequency Decision Matrix (HDM)
- Refusal Code Level Data Store (RCL
- Year-of-Birth Range Data Store (YR)
- Refusal Code Level/Year-of-Birth Range Data Store (RLYOB)
- Country-of-Birth Proximity Data Store (COBPROX)
- Hispanic Country-of-Birth Category Data Store (HCOB)

4.4. HISPANIC TITLE/AFFIX/QUALIFIER DATA STORE DECOMPOSITION

Because the HNA-E design is viewed as an independent sub-program of the CLASS-E system, the Hispanic Title/Affix/Qualifier Data Store is presented here as a separate table. It is strongly suggested, however, that CLASS-E support one TAQ Data Store in which the cultural affinity of each TAQ segment is indicated. This will reduce table maintenance and will provide a global picture of the handling of TAQs.

4.4.1. **Identification**

This data store is known as the Hispanic Title/Affix/Qualifier (TAQ) Data Store (HTD).

4.4.2. **Type**

4.4.2.1. The HTD is a data store that contains the Hispanic-specific Title, Affix and Qualifier segments with additional information about the disposition of the items.

4.4.2.2. The HTD will be accessed by the Hispanic Name Preprocessor (HNP) and the Hispanic Filter and Sorter (HFS).

4.4.2.3. The format of the HTD will be

Figure 53: Format: Hispanic TAQ Data Store (HTD)

DATA FIELD	DATA TYPE	FIELD SIZE	DATA RANGE/VALUE
ID_NO	integer	4	0...9999
TAQ_FORM	character	15	alphabetics
TAQ_TYPE	character	1	T, I, P, S Q
DELETE	integer	1	1, 0 (True, False)
DISREGARD	integer	1	1, 0 (True, False)
REMOVE	integer	1	1, 0 (True, False)

4.4.2.4. Definitions

4.4.2.4.1. ID_NO: a unique, arbitrary number that identifies the TAQ segment.

4.4.2.4.2. TAQ_FORM: the string that represents the TAQ; the TAQ_FORM may be a multipart string (i.e., a string that includes internal white space).

4.4.2.4.3. TAQ_TYPE: an indicator of the kind of TAQ segment present: a title (T), prefix (P), infix (I), suffix (S), or qualifier (Q).

4.4.2.4.4. DELETE:

4.4.2.4.4.1. The segment is to be removed from all further consideration in the name search process.

4.4.2.4.4.2. The segment is referenced in the filtering process.

4.4.2.4.4.3. The segment is not removed from the original record and is returned with the record to the user.

4.4.2.4.4.4. True (1) or False (0) indicates whether or not this function is to apply to the segment(s) under consideration.

4.4.2.4.5. DISREGARD:

4.4.2.4.5.1. The segment is to be removed from further consideration in the name search process but will undergo special evaluation in the filtering process. It will be returned with the record to the user.

4.4.2.4.5.2. True (1) or False (0) indicates whether or not this function is to apply to the segment(s) under consideration.

4.4.2.4.6. REMOVE:

4.4.2.4.6.1. The segment occurs attached to the name stem.

4.4.2.4.6.2. The conjoined TAQ will be separated from a base name segment. (See Section 3.5.4.4.3).

4.4.2.4.6.3. True (1) or False (0) indicates whether or not this function is to apply to the segment(s) under consideration.

4.4.2.4.6.4. The separated segment will also be marked for DELETE/DISREGARD treatment.

4.4.3. **Purpose**

Peripheral elements (Titles, Affixes, and Qualifiers) in names do not contribute as much to the name evaluation as does the name stem. Identifying and removing these elements in the name processing component is important. They do, however, contribute to the overall value of a name when determining the proximity of one name to another. They will therefore contribute some value to the filtering and sorting processes.

4.4.4. **Function**

The HTD serves as a repository for all TAQ values and for the treatment that each will be subjected to.

4.5. HIGH FREQUENCY SURNAME TYPE DATA STORE DECOMPOSITION

4.5.1. **Identification**

4.5.1.1. This data store is known as the High Frequency Surname Type Data Store (HFST).

4.5.1.2. This data store could be merged with the High Frequency Surname Variant Data Store (HFSV).

4.5.1.2.1. The ID_NO would be different in the HFSV and would serve as a unique identifier for each entry.

4.5.1.2.2. The set of HFSN_TYPEs, with no variants, would be derivable from the HFSN_VARs with a DI_VAL equal to 1.00.

4.5.1.2.3. The SET_ID of the HFST and HFSV would be the same.

4.5.2. **Type**

4.5.2.1. The HFST data store consists of the 500 most frequently occurring HF SN segment types (i.e., unique occurrence).

4.5.2.2. The HFST will be accessed by the Hispanic Surname Segmenter, Hispanic Segment Positioner, and the Frequency Path Director modules.

4.5.2.3. The format of the HFST will be

Figure 54: Format: High Frequency Surname Type Data Store (HFST)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
ID_NO	integer	4	1...9999
HFSN_TYPE	character	24	alphabetics
SET_ID	integer	4	1...9999

Figure 55: Example: Piece of HFST

ID_NO	HFSN_TYPE	SET_ID
0001	GARCIA	0001
0002	RODRIGUEZ	0002
0003	HERNANDEZ	0003
0004	LOPEZ	0004
0005	MARTINEZ	0005
0006	GONZALEZ	0006
0007	PEREZ	0007
0008	SANCHEZ	0008
0009	RAMIREZ	0009
0010	GOMEZ	0010
0011	...	0011

4.5.2.4. Definitions

4.5.2.4.1. ID_NO will be a unique numerical identifier for each of the HF SN segments, HFSN_TYPEs.

4.5.2.4.2. HFSN_TYPE will contain a unique character string that represents one of the 500 most frequently occurring Hispanic surname stems.

4.5.2.4.3. SET_ID will be the unique identifier for the *set* of variants of the HFSN_TYPE and will be used as the HFSN_KEY.

4.6. HIGH FREQUENCY SURNAME VARIANT DATA STORE DECOMPOSITION

4.6.1. Identification

4.6.1.1. This data store is known as the High Frequency Surname Variant Data Store (HFSV).

4.6.1.2. This data store will be updated in real time as variants qualify for inclusion in the data store. (See Section 3.12.4.38)

4.6.1.3. This data store could be merged with the HFST (See Section 4.5.1.2.)

4.6.2. Type

4.6.2.1. The HFSV is a data store that consists of a HFSN_TYPE segment with a variant of that segment and a value that represents the degree of digraph proximity of the HFSN_TYPE and its variant.

4.6.2.2. The HFSV is a data store that will have between 75,000 and 100,000 rows.

4.6.2.3. A name segment may be the variant of more than one HFSN_TYPE.

4.6.2.4. The HFSV will be accessed by the High Frequency Processor and Low Frequency Processor.

4.6.2.5. The format of the HFSV will be

Figure 56: Format: High Frequency Surname Variant Data Store (HFSV)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
ID_NO	integer	6	000000…999999
HFSN_VAR	character	24	alphabetics
SET_ID	integer	4	0000…9999
DI_VAL	decimal	4	0.00...1.00

Figure 57: Example: High Frequency Surname Variant Data Store

ID_NO	HFSN_VAR	SET_ID	DI_VAL
032711	PEREZ	0007	1.00
032712	PERES	0007	0.67
032713	PEREZA	0007	0.77
016976	GOMEZ	0010	1.00
016977	GOMES	0010	0.67
016978	BOMEZ	0010	0.67

4.6.2.6. Definitions

4.6.2.6.1. ID_NO will be a unique numerical identifier for each HFSN_VAR entry.

4.6.2.6.2. HFSN_VAR will contain a character string that has been determined to be a variant of the HFSN_TYPE with which it is associated. A HFSN_VAR may be a variant of one or more of the HFSN_TYPEs.

4.6.2.6.2.1. A variant is defined as a name stem that shares a sufficient number of digraphs (strings of two characters) with the HFSN_TYPE to pass a pre-determined threshold.

4.6.2.6.2.2. A HFSN_TYPE can be obtained from the HFST or from the HFSV as a HFSN_VAR with a DI_VAL equal to 1.00.

4.6.2.6.3. DI_VAL is a two-place decimal value that represents the proximity of the HFSN_TYPE and the HFSN_VAR.

4.6.2.6.3.1. The DI_VAL is a calculation derived from the shared digraphs (strings of two characters) of the HFSN_TYPE and the HFSN_VAR associated with it.

4.6.2.6.3.2. The calculation is determined in the following way:

4.6.2.6.3.2.1. The digraphs are identified for each name stem, the HFSN_TYPE (Comparand #1) and the HFSN_VAR (Comparand #2).

4.6.2.6.3.2.1.1. Each pair of alphabetic characters is identified: GOMEZ → GO / OM / ME / EZ

4.6.2.6.3.2.1.2. A digraph is also formed of the initial boundary (#) and the first alphabetic character: GOMEZ → #G.

4.6.2.6.3.2.1.3. A digraph is also formed of the final alphabetic character and the final boundary (#): GOMEZ → Z#.

4.6.2.6.3.2.2. The number of shared digraphs is calculated; a digraph may match one digraph only.

4.6.2.6.3.2.3. The number of shared digraphs is multiplied by 2 and divided by the total number of digraphs in comparand #1 added to the total number of digraphs in comparand #2.

4.6.2.6.3.2.4. The formula is:

2 * d / a + b, where *d* = the total number of shared digraphs; where *a* = the total number of digraphs in Comparand #1 and where *b* = the total number of digraphs in Comparand #2.

Figure 58: Example: Digraph Evaluation of Two Comparands

COMPARANDS	DIGRAPHS	SHARED DIGRAPHS (d)	DI_VAL
COMPARAND #1: DOMINGUEZ	#D DO OM MI IN NG GU UE EZ Z#	#D DO OM MI IN UE	2*d / a + b = 12 / 20
COMPARAND #2: DOMINQUES	#D DO OM MI IN NQ QU UE ES S#	= 6	.60

4.6.3. **Purpose**

4.6.3.1. For HNA-E to be an effective retrieval system, it must be able to retrieve variants of query names. The impact on system performance can be dramatic, however, if traditional matching techniques are used to identify variant names. By assigning variants to the same set and recording their digraph value, querying a HF surname will result in the direct retrieval of variant records and their digraph values.

4.6.3.2. The HFSV also serves as a resource for identifying which HF surnames are related to a LF surname.

4.6.4. **Function**

The HFSV Data Store will be dynamically updated. (See Section 3.12.4.38 for details.)

4.7. HIGH FREQUENCY DECISION MATRIX DATA STORE

4.7.1. **Identification**

This data store is known as the High Frequency Decision Matrix (HDM).

4.7.2. **Type**

4.7.2.1. The HDM is a data store that will provide criteria for database record retrieval for query records with HF name segments.

4.7.2.2. It will be accessed by the High Frequency Processor (HFP).

Figure 59: Format: Hispanic Decision Matrix (HDM)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
QUERY SN FORMAT	character	2	A, B
DATABASE SN FORMAT	character	2	A, B, C
YOB RANGE (YOB#)	integer	1	0...6
REFUSAL CODE LEVEL (RL#)	integer	1	0...4
RECORD GENDER (RGNDR)	character	3	M, F, U

Figure 60: Example: Hispanic Decision Matrix (HDM) (Values for example only)

	Single-Segment SN				Two-Segment SN							
QUERY SN FORMAT	A	A	A		AB	AB	AB	AB	AB	AB	AB	AB
DATABASE SN FORMATS	A	AB	BA		AB	BA	A	B	AC	CA	CB	BC
YOB#	5	5	2		5	4	4	2	2	0	0	0
RL#	4	4	3		4	4	4	1	1	0	0	0
RGNDR	MFU	MFU	MFU		MFU	MFU	MFU	MFU	FU	MFU	MFU	MFU

4.7.3. Definitions

4.7.3.1. QUERY SN FORMAT: is a character string that is an abstract representation of the query SN. Each segment is represented by a single character, the leftmost A, the next B. The sequence also represents the position of the segment.

4.7.3.2. DATABASE SN FORMAT: is a character string that is an abstract representation of the possible and acceptable variations in the query SN which are relevant to the QUERY SN FORMAT and which will be retrieved from the database, given the conditions stipulated in the YOB RANGE (YOB#), REFUSAL CODE LEVEL (RL#) and RECORD GENDER (RGNDR). Each segment is represented by a single character.

4.7.3.2.1. If the character is the same as a character in the QUERY SN FORMAT, it represents the same SN Key.

4.7.3.2.2. If the character is different from a character in the QUERY SN FORMAT, it represents a different SN Key.

4.7.3.2.3. If the character is in the same relative position as that in the query SN, it represents the same position in the SN string.

4.7.3.2.4. If the character in not in the same relative position as that in the query SN, it represents a different (out-of) position in the SN string.

4.7.3.3. YOB RANGE (YOB#): is an integer that represents a YOB range specified in the YOB RANGE (YR) Data Store. (N.B. In this scheme, YOB# integer does not represent the year range itself. It refers to a table that specifies that YOB2, for example, represents an exact year-of-birth and that YOB3 represents a range of 1 year on either side of the query year (for a range total of 3 years).)

4.7.3.4. REFUSAL CODE LEVEL (RL#): is an integer that represents a Refusal Code Level specified in the REFUSAL CODE LEVEL Data Store. (N.B. In this scheme, this number represents a set of Refusal Codes that has a pre-determined degree of seriousness. The number given here does not signal the Refusal Code itself. The number is expanded in the Refusal Code Level Data Store, where 0, for example, might represent a 00 Refusal Code.)

4.7.3.5. RECORD GENDER (RGNDR): is a set of up to three characters that represent the required Record Gender of the database record.

4.7.4. **Purpose**

Many Hispanic surnames occur with very high frequency; they also generally have at least two segments. Any retrieval system that captures only one of these names will have an inordinately high recall. Many of these records will not be at all relevant to the query record. Special treatment of high frequency names must entail some method of reducing the number of irrelevant records retrieved from the database. The HDM provides the information about how to delimit the records that will be retrieved from the database. A reduction in the recall will reduce post-processing time.

4.7.5. **Function**

The HDM is a data store that consists of qualifying and delimiting criteria.

4.7.5.1. Qualifying criteria will be the number of SN segments, SN content, and SN segment positions.

4.7.5.2. Delimiting criteria will be Year-of-Birth (YOB) Range (YR), Refusal Code (RC) Level (RL) and Record Gender (RGNDR).

4.7.5.2.1. The qualifying criteria will produce a set of SN formats to retrieve from the database.

4.7.5.2.2. The delimiting criteria will specify the YOB range, maximum RC Level for each of the SN formats and Record Gender limitations, if any.

4.8. HISPANIC GIVEN NAME TYPE DATA STORE DECOMPOSITION

4.8.1. **Identification**

4.8.1.1. This data store is known as the Hispanic Given Name Type Data Store (HGT).

4.8.1.2. This data store could be merged with the High Frequency Given Name Variant Data Store (HFGV).

4.8.1.2.1. The ID_NO would be different in the HFGV and would serve as a unique identifier for each entry.

4.8.1.2.2. The set of HFGN_TYPEs, with no variants, would be derivable from the HFGN_VARs with a DI_VAL equal to 1.00.

4.8.1.2.3. The SET_ID of the HFGT and HFGV would be the same.

4.8.2. **Type**

4.8.2.1. The HGT data store will consist of up to ten thousand entries.

4.8.2.2. The HGT will be accessed by the Hispanic Gender Identifier, Frequency Path Director (FPD), the High Frequency Processor.

4.8.2.3.The HGT will have the following format:

Figure 61: Format: Hispanic Given Name Type Data Store (HGT)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
ID_NO	integer	4	0000...9999
GN_TYPE	character	24	alphabetics
SET_ID	integer	3	001...999
HI_FREQ	integer	1	1, 0 (True, False)
GNDR	character	1	M, F, U

Figure 62: Example: Hispanic Given Name Type Data Store (HGT)

ID_NO	GN_TYPE	SET_ID	HI_FREQ	GNDR
0001	JOSE	0001	1	M
0002	MARIA	0002	1	F
0003	JUAN	0003	1	M
0004	LUIS	0004	1	M
0005	ANTONIO	0005	1	M
0006	CARLOS	0006	1	M
0007	JESUS	0007	1	M
0008	MANUEL	0008	1	M
0009	FRANCISCO	0009	1	M
0010	JORGE	0010	1	M
0011	...	0011		...
2367	DAGOBERTO	0000	0	M

4.8.2.4. Definitions

4.8.2.4.1. ID_NO: is an integer that is a unique numerical identifier for each of the GN_TYPEs.

4.8.2.4.2. GN_TYPE: is a a character string that represents one of up to ten thousand Hispanic given name stems.

4.8.2.4.2.1. A HFGN_TYPE is a GN_TYPE whose HI_FREQ value is 1 (True).

4.8.2.4.3. SET_ID: is an integer that is the numerical identifier for the *set* of related variants of the GN_TYPE that is HF.

4.8.2.4.3.1. The SET_ID will serve as the HFGN_KEY.

4.8.2.4.3.2. Not every entry in the HGT will have a unique SET_ID; a distinct SET_ID is reserved for those GN_TYPEs where HI_FREQ is True (1).

4.8.2.4.4. HI_FREQ: is an integer (1, 0/True, False) that indicates if the GN_TYPE is or is not a HF GN segment.

4.8.2.4.4.1. The frequency of all GN_TYPEs will be specified.

4.8.2.4.4.2. True (1) will indicate a HF segment.

4.8.2.4.4.3. False (0) will indicate a LF segment.

4.8.2.4.5. GNDR: is a single character value that indicates the gender of the GN_TYPE.

4.8.2.4.5.1. If the name is predictably female, the value will be F.

4.8.2.4.5.2. If the name is predictably male, the value will be M.

4.8.2.4.5.3. If the name is ambiguous or unknown, the value will be U.

4.8.3. **Purpose**

The HGT provides information about Hispanic given name segments. It indicates the frequency of the segments, their gender and the set of names of which they are the parent.

4.8.4. **Function**

The HGT serves as a resource for Hispanic Gender Identifier and the High Frequency Processor.

4.9. HIGH FREQUENCY GIVEN NAME VARIANT DATA STORE DECOMPOSITION

4.9.1. **Identification**

This data store is known as the High Frequency Given Name Variant Data Store (HFGV).

4.9.2. **Type**

4.9.2.1. The HFGV will be accessed by the High Frequency Processor.

4.9.2.2. The HFGV will have about 60,000 to 90,000 rows.

4.9.2.3. The HFGV will have the following format:

Figure 63: Format: High Frequency Given Name Variant Data Store (HFGV)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
ID_NO	integer	5	00000...99999
HFGN_VAR	character	24	alphabetics
SET_ID	integer	4	0000...9999
DI_VAL	decimal	4	0.00...1.00

Figure 64: Example: Piece of HFGV

ID_NO	HFGN_VAR	SET_ID	DI_VAL
00001	JOSE	0001	1.00
00002	JOSEA	0001	0.73
00003	JOSSE	0001	0.73
00004	MARIA	0002	1.00
00005	MIRIA	0002	0.67
00006	MIRIAM	0164	1.00
00007	MIRIA	0164	0.77

4.9.2.4. Definitions

4.9.2.4.1. ID_NO: is a unique numerical identifier for each HFGN_VAR entry in the HFGV data store.

4.9.2.4.2. HFGN_VAR: is character string that represents a GN segment that is a digraph variant of the HFGN_TYPE (HFGN_VAR whose DI_VAL = 1.00).

4.9.2.4.3. SET_ID: is a unique identifier of the set of GN segments that are variants of the same HFGN_TYPE.

4.9.2.4.4. DI_VAL: is a two-place decimal value that indicates the digraph relationship between the HFGN_VAR and its parents HFGN_TYPE.

4.9.3. **Purpose**

The HFGV is a resource for defining the given name segments that will be stored with records added to the database. Storage of information about variant relations will speed retrieval and the filtering process.

4.9.4. **Function**

The HFGV will be accessed by the HFP to assign keys to given name segments on record add and query.

4.10. LOW FREQUENCY SURNAME TYPE DATA STORE DECOMPOSITION

4.10.1. Identification

This data store is known as the Low Frequency Surname Type Data Store (LFST).

4.10.2. Type

4.10.2.1. The LFST is a data store of LF keys.

4.10.2.2. The LFST will have about 900,000 to 1 million rows.

4.10.2.3. The LFST will have the following format:

Figure 65: Format: Low Frequency Surname Type Data Store (LFST)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
ID_NO	integer	6	000001...999999
LFSN_TYPE	character	24	alphabetics
LFDIKEY	character	3	alphanumerics; char + char + #

Figure 66: Example: Piece of LFST

ID_NO	LFSN_TYPE	LFDIKEY
000001	AALVAREZ	AA1
000001	AALVAREZ	AA2
000001	AALVAREZ	AL2
000001	AALVAREZ	AL1
000001	AALVAREZ	AL3
000001	AALVAREZ	LV3
000001	AALVAREZ	LV2
000001	AALVAREZ	LV4
000001	AALVAREZ	VA4
000001	AALVAREZ	VA3
000098	BARRIOS	BA1
000098	BARRIOS	BA2
...		

4.10.2.4. Definitions:

4.10.2.4.1. ID_NO: is an arbitrary numerical reference to each LFSN_TYPE. The ID_NO will serve as the DI_KEY.

4.10.2.4.2. LFSN_TYPE: is the unique low frequency name segment as it occurs in the database; if there are multiple occurrences of the same name, they are represented by one entry, hence the term "type."

4.10.2.4.3. LFDIKEY: is a string of alphanumeric characters that represents one digraph and its actual or derived position.

4.10.2.4.3.1. Up to ten LFDIKEYs will be associated with each LFSN_TYPE.

4.10.2.4.3.2. An LFDIKEY is name-specific, so the same key may appear with other LFSN_TYPEs, in which case it will have a different ID_NO.

4.10.2.4.3.3. A LFDIKEY is

1) a digraph formed from the LF SN segment beginning with the leftmost character and its position (Base Key) and
2) a positional variant on that digraph key (Position Key).

4.10.2.4.3.4. Positional information will be associated with each digraph.

4.10.2.4.3.5. To form a key, begin with the leftmost character and generate four digraph keys (Base Key) from the five leftmost characters of the LF SN segment. The first two characters form a digraph, the second and third characters form a digraph, the third and fourth characters form a digraph and the fourth and fifth characters form a digraph. Positional information (Positions 1, 2, 3, 4) will be included.

4.10.2.4.3.6. Generate, from the Base Keys, up to six additional Position Keys; the position keys have the same characters as the Base Keys but contain different positional information. A maximum of ten keys (Base + Position) will be generated.

4.10.2.4.3.6.1. Produce a Position Key on the first Base Key with Position 2.

4.10.2.4.3.6.2. Produce Position Keys on the second Base Key with Position 1 and Position 3.

4.10.2.4.3.6.3. Produce Position Keys on the third Base Key with Position 2 and Position 4.

4.10.2.4.3.6.4. Produce a Position Key on the fourth Base Key with Position 3. No Position Key is generated for Position 5 because the maximum of 10 keys has been reached.

4.10.3. **Purpose**

The LFST provides information that will limit the search of database records. Preprocessing of name types allows identification of relevant name segments without having to examine database records directly.

4.10.4. **Function**

The LFST will be accessed by the LFP.

4.11. HISPANIC CHARACTER DATA STORE

4.11.1. **Identification**

This data store is known as the Hispanic Character Data Store (HCD).

4.11.2. **Type**

4.11.2.1. The HCD is a data store of all characters in Hispanic names and their predictable variants.

4.11.2.2. The format of the HCD will be:

Figure 67: Format: Hispanic Character Data Store (HCD)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE
SET_ID	integer	3	000...999
CHAR	character	1	alphabetics
CHAR_VAR	character	1	alphabetics

Figure 68: Example: Piece of HCD

SET_ID	CHAR	CHAR_VAR
001	B	B
001	B	V
002	S	S
002	S	Z
004	C	C
004	C	S
...		
037	F	F
052	K	K
078	M	M
078	M	N
...		

4.11.2.3. Definitions

4.11.2.3.1. SET_ID: is an arbitrary numerical that represents the set of characters that vary with one another. The SET_ID will be the GN_INIT Key.

4.11.2.3.2. CHAR: is a single alphabetic character. Every alphabetic character will be represented. The CHAR is the type of character, which may or may not have variants (CHAR_VAR).

4.11.2.3.3. CHAR_VAR: is a single alphabetic character that may or may not vary predictably with other characters in written Spanish. A single character may participate in more than one set.

4.11.3. **Purpose**

Retrieval of records with HF SN segments from the database will be limited by the initial of the GN segments. For the retrieval to be sufficiently robust, however, the system must allow for some variation in the GN initials. The HCD indicates variations on initials.

4.11.4. **Function**

The HCD will be accessed by the HFP and will provide the source of the GN_INIT Keys that are to be generated for HF searches.

4.12. TAQ FILTER DATA STORE DECOMPOSITION

4.12.1. **Identification**

This data store is known as the TAQ Filter Data Store (TF).

4.12.2. **Type**

4.12.2.1. This TF will be accessed by the Hispanic Filter and Sorter and provides parameter factors for matching TAQ DISREGARD tags during record filtering.

4.12.2.2. The format of the TF follows:

Figure 69: Format: TAQ Filter Matrix Design

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
TAQDIS#1	character	8	alphabetics	TAQ_DISREGARD ITEM
TAQDIS#2	character	8	alphabetics	TAQ_DISREGARD ITEM
TF_VALUE	decimal	4	0.00...1.00	Various (TBD)

Figure 70: Example: Piece of TF (Values are for example only)

TAQDIS#1	TAQDIS#2	TF_VALUE
DE	DE	1.00

DE	DEL	0.90
DE	DE LOS	0.90
DE	LOS	0.75
DE	SAN	0.75
DE	LA	0.75
DEL	DEL	1.00
DEL	DE LOS	0.75
DEL	LOS	0.65
DEL	LA	0.85
DEL	SAN	0.50
DE LOS	DE LOS	1.00
DE LOS	LOS	0.90
DE LOS	SAN	0.50
DE LOS	LA	0.50
SAN	SAN	1.00
SAN	LOS	0.50
SAN	LA	0.50
LOS	LOS	1.00
LOS	LA	0.85
LA	LA	1.00
...		

4.12.2.3. Definitions

4.12.2.3.1. TAQDIS#1: is the TAQ DISREGARD segment that occurs in one or the other (different) of the comparands.

4.12.2.3.2. TAQDIS#2: is the TAQ DISREGARD segment that occurs in one or the other (different) of the comparands.

4.12.2.3.3. TF_VALUE: is the factor that will be used to adjust the SN_VAL or GN_VAL if the TAQDIS#1 and TAQDIS#2 are present in the comparands.

4.12.3. **Purpose**

Hispanic names often have peripheral name elements. Some of these make up a segment of the name, the TAQ values identified in the TF. Their relative value, however, varies. Some of them cannot cooccur, some have opposite meanings, so it is necessary to identify their relative value when they are contrasted with one another.

4.12.4. **Function**

The TF provides the resources for the HFS to determine the relative value of TAQs that occur in two comparands.

4.13. HISPANIC PARAMETER DATA STORE DECOMPOSITION

4.13.1. **Identification**

This module is known as the Hispanic Parameter Data Store (HPD).

4.13.2. **Type**

4.13.2.1. The HPD is a data store that will be accessed by the Filter Component of the Hispanic Filter and Sorter (HFS).

4.13.2.2. The HPD is a parameter table that will be accessible to the user and whose cell values will be determined through testing and comparative evaluation.

4.13.2.3. The HPD has the following format:

Figure 71: Format: Hispanic Parameter Data Store (HPD)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
PARM_NAME	character	6	alphabetics	SNTHR, GNTHR, OPSVAL, OPGVAL, INITSN, INITGN, RGNDR, TAQASN, TAQAGN, TAQXSN, TAQXGN, RL#, YOB#, COB#, etc.
PARM_VAL	decimal	4	0.00…1.99	Various (TBD)

Figure 72: Example: HPD (Values are for example only.)

PARM_NAME	PARM_VAL
SNTHR	0.60
GNTHR	0.65
LFDIKEY THRESHOLD	0.57

DI_VAL THRESHOLD	0.63
HFGV THRESHOLD	0.65
HFSV THRESHOLD	0.65
OPGVAL	0.60
OPSVAL	0.60
ASVAL	0.65
AGVAL	0.65
INITSN	0.85
INITGN	0.85
INITNM	0.80
RGNDR	0.65
TAQASN	0.90
TAQAGN	0.90
TAQXSN	0.85
TAQXGN	0.85
RL0	1.20
RL1	1.15
RL2	1.10
RL3	1.05
RL4	1.00
YOB0	1.30
YOB1	1.25
YOB2	1.20
YOB3	1.15
YOB4	1.10
YOB5	1.05
YOB6	1.00
COB1	1.20
COB2	1.15
COB3	1.10
COB4	1.00
COB5	0.95

4.13.2.4. The values provided are for example only and do not necessarily represent the PARM_VALs to be used for the parameters.

4.13.3. **Purpose**

The HPD is a data store that allows easy access to adjustable thresholds for record qualification, to thresholds for data store updates, and to parameters that contribute to the determination of the name scores (SN_VAL, GN_VAL) and to the Composite Score of two record comparands.

4.13.4. **Function**

The HP functions as an independent data store with thresholds needed by the LFP and all the parameters needed by the HFS during the filtering process.

4.14. REFUSAL CODE CATEGORY DATA STORE DECOMPOSITION

4.14.1. **Identification**

This data store is known as the Refusal Code Level Data Store (RCL).

4.14.2. **Type**

4.14.2.1. It is recommended that the RCL be a parameter file, which can be accessed by the client so RC categories can be added to or changed with ease.

4.14.2.2. The RC data store will provide a list of the Refusal Codes and its Refusal Category, which is an indication of the level of seriousness of each Refusal Code.

4.14.2.3. The RCL will be referred to by the Hispanic Decision Matrix (HDM) and by the LFP and Hispanic Filter and Sorter.

4.14.2.4. The RCL has the following format:

Figure 73: Format: Refusal Code Level Data Store (RCL)

DATA FIELD	DATA TYPE	FIELD SIZE	DATA VALUE
REFUSAL CODE	alphanumerics	3	Standard Refusal Codes
REF_CAT	alphanumerics	3	RL0, RL1, RL2, RL3, RL4

Figure 74: Example: RCL (REF_CATs for example only)

REFUSAL CODE	REF_CAT
00	RL0
23	RL1
6C	RL2
07	RL3
G	RL4

4.14.2.5. Definitions

4.14.2.5.1. REFUSAL CODE: indicates each Visa Refusal Code (Codes and their Refusal Level (see VALUE) are for example only; they do not represent the complete list nor the accurate assignment of a Refusal Code to a Refusal Level).

4.14.2.5.2. REF_CAT: The RL# will appear in the form RL1, RL2, etc.

4.14.2.5.2.1. RL# is the Refusal Category to which a particular Refusal Code has been assigned. The Visa Office will assign Refusal Codes to one of 4 categories: RL1, RL2, RL3, RL4; RL0 is reserved for the Refusal Code 00. (The current distinction among Refusal Codes is a binary one: serious and non-serious. Assignment of Refusal Codes to more groups has not yet been done; the consequence is that one or more of these categories may not currently have a distinct value.) The RL# occurs in ascending order, from most serious to least serious Refusal Code. The RL# will be linked to a Year-of-

Birth Code (see Section 4.16) to determine the relevant subsets of records to be searched.

4.14.2.5.2.2. **RC0** refers to the Refusal Code 00

4.14.2.5.2.3. **RC1** refers to all Refusal Codes that have been designated as Type 1 Serious RC 1, i.e., the most serious, excluding 00.

4.14.2.5.2.4. **RC2** refers to all Refusal Codes that have been designated as Type 2 Serious RC, i.e., serious but less serious than RC0 and RC1.

4.14.2.5.2.5. **RC3** refers to all Refusal Codes that have been designated as Type 1 Non-Serious RC. These codes are less serious than the RC0, RC1 and RC2 codes.

4.14.2.5.2.6. **RC4** refers to Refusal Codes that have been designated as Type 2 Non-Serious. These codes are the least serious codes, less serious than the RC0, RC1, RC2 and RC3 codes.

4.14.3. Purpose

It has long been desirable to make more granular distinctions among the Refusal Codes. For many years, DOS has maintained a distinction between serious and non-serious codes; these different categories were correlated with different YOB search ranges. However, a mechanism for making greater distinctions will provide greater flexibility in delimiting the set to be retrieved during the first stage of record analysis, especially for Hispanic high frequency names, where more restricted retrievals are highly desirable. The introduction of five refusal code levels also provides the opportunity to correlate more year-of-birth ranges to the refusal code levels.

4.14.4. Function

The RCL provides information needed for the evaluation of record proximity in the Hispanic filtering process and contributes to the delimitation of database records retrieved through the RLYOB Data Store.

4.15. YEAR-OF-BIRTH RANGE DATA STORE DECOMPOSITION

4.15.1. Identification

This data store is known as the Year-of-Birth Range Data Store (YR).

4.15.2. **Type**

4.15.2.1.It is recommended that the YR be a parameter file, which can be accessed by the client so YOB ranges can be set. Alternatively, it could be represented as a system parameter whose value(s) are set in an .ini file.

4.15.2.2. The YR will define the YOB ranges that will be associated with a Refusal Level (see Section 4.16).

4.15.2.3. This data store has the following format:

Figure 75: Format: Year-of-Birth Range Data Store (YR)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE	DATA DEFINITION
YOB0	integer	1	0	exact date of birth
YOB1	character	1	A	exact year, inverted month and day
YOB2	character	1	B	exact year of birth
YOB3	integer	2	1...99	narrow year of birth range
YOB4	integer	2	1...99	standard year of birth range
YOB5	integer	2	1...99	wide year of birth range
YOB6	integer	2	1...99	unlimited year of birth range

4.15.2.4. Definitions

4.15.2.4.1. YOB# is the Year-of-Birth Range category whose value indicates the year-of-birth range to be searched. The year-of-birth VALUE indicates the search range, that is, the number of years on either side of a given year-of-birth to be searched. For example, if the input year is 1962 and YOB3 range is 4, the search will cover a range of nine years, 1958-1966. The range includes the full year, so all of 1958 and all of 1966.

4.15.2.4.1.1. There are seven YOB# categories, YOB0, YOB1, YOB2, YOB3, YOB 4, YOB5, YOB6.

- **YOB0** is a single integer that refers to an exact month, day, year of birth. If YOB0 is specified, the system must be able to match the month, day and year of the Date of Birth of an input record and a database record.
- **YOB1** is a single character (A) that refers to an exact year-of-birth with the month and day inverted.
  - If YOB1 is specified, the system must be able to match the year of Date of Birth and an inverted month and day (DEC 03 → MAR 12) of the input record and the database record.
  - YOB1 will be relevant to the Hispanic Filter and Sorter, but may not function as a search

parameter since the value would be subsumed in YOB2.

- **YOB2** is a single character (B) that refers to an exact year-of-birth. If YOB2 is specified, the system must be able to match the year of the Date-of-Birth of an input record and a database record.
- **YOB3** is a one- or two-place integer (1...99) that refers to a narrow year-of-birth range. Narrow year-of-birth range is usually defined as 1 year (for a search range of 3 years).
- **YOB4** is one- or two-place integer (1...99) that refers to a standard year-of-birth range. Standard year-of-birth range is usually defined as 3 years (for a search range of 7 years).
- **YOB5** is a one- or two-place integer (1...99) that refers to a wide year-of-birth range. Wide year-of-birth range is usually defined as 5 years (for a search range of 11 years).
- **YOB6** is a one- or two-place integer (1...99) that refers to an unlimited or extremely wide year-of-birth range. Unlimited year-of-birth range would be set sufficiently high to include all (or all desired) years-of-birth in the database (e.g., 50).

4.15.3. Purpose

This YR provides a greater granularity in the year-of-birth range and, therefore, greater flexibility in delimiting the set to be retrieved during the first stage of record analysis. The correlation of five refusal code levels to different year-of-birth ranges will help to delimit the number of records to be searched and to define the more valuable set of records. For the Hispanic processor, greater precision in the year-of-birth range is especially important in the High Frequency Processor where more restricted retrievals are highly desirable.

4.15.4. Function

4.15.4.1. The YR permits greater granularity in the Date-of-Birth types related to the system.

4.15.4.2. The YR will be accessed by the Refusal Code Level/YOB Range Data Store, which will limit the retrieval range in the Hispanic Search Engine.

4.15.4.3. The YR data store will define the YOB ranges referred to in the Hispanic Decision Matrix (HDM).

4.15.4.4. The YR will contribute to the Hispanic Filter and Sorter to contribute information to the composite score.

4.16. REFUSAL CODE LEVEL / YOB RANGE DATA STORE MODULE DECOMPOSITION

4.16.1. **Identification**

This data store is known as the Refusal Code Level/YOB Range Data Store (RLYOB).

4.16.2. **Type**

4.16.2.1. The RLYOB is a matrix that merges the values in the Refusal Code Level (RCL) Data Store and the Year-of-Birth Range (YR) Data Store.

4.16.2.2. For each Refusal Level (RL), a Year-of-Birth (YOB) Range is specified.

4.16.2.2.1. Only one YOB Range for each RL is permitted.

4.16.2.2.2. The same YOB Range may apply to more than one RL.

4.16.2.3. The RLYOB has the following format:

Figure 76: Format: Refusal Level/Year-of-Birth Range Data Store (RLYOB)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
RL#	character	3	RL0...4	RL0, RL1, RL2, RL3, RL4
YOB#	character	4	YOB0...6	YOB0, YOB1, YOB2, YOB3, YOB4, YOB5, YOB

Figure 77: Example: RLYOB Data Store

RL#	YOB#
RL0	YOB5
RL1	YOB4
RL2	YOB3
RL3	YOB3
RL4	YOB2

4.16.2.4. Definitions:

4.16.2.5. RL#: is a character string that indicates the Refusal Level of the Refusal Code.

4.16.2.6. YOB#: is a character string that indicates the Date-of-Birth Range Category of the comparands.

4.16.3. **Purpose**

Retrieval of records from the database should be delimited by a relationship between the Refusal Code Level and the Year-of-Birth Range. It will restrict the number of records to be reviewed.

4.16.4. **Function**

The RLYOB is a resource for the Hispanic Search Engine to delimit the LF records retrieved from the database.

4.17. COUNTRY-OF-BIRTH PROXIMITY DATA STORE

4.17.1. Identification

This module is known as the Country-of-Birth Proximity Data Store (COBPROX).

4.17.2. Type

4.17.2.1. The COBPROX is a data store whose cells contain a decimal value that reflects the degree of relationship between the country represented on two country-of-birth.

4.17.2.2. The COBPROX has the following format:

Figure 78: Design: COBPROX Data Store

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
COB#1	character	4	alphabetics	COB Code
COB#2	character	4	alphabetics	COB Code
COBVAL	decimal	4	0.00...1.00	Various

Figure 79: Example: Piece of COBPROX Data Store (COBVAL for example only)

COB#1	COB#2	COBVAL
AGS	AGS	1.00
AGS	GRBR	0.05
AGS	VTNM	0.05
AGS	MORO	0.05
AGS	SYR	0.05
ALG	ALG	1.00
ALG	MORO	0.85
ALG	GRBR	0.05
ALG	VTNM	0.05
MORO	MORO	1.00
MORO	GRBR	0.05
MORO	VTNM	0.05
GRBR	GRBR	1.00
GRBR	VTNM	0.05
VTNM	VTNM	1.00

4.17.2.3. Definitions:

4.17.2.3.1. COB#1: is the 4-character Country-of-Birth Code of one of the comparands.

4.17.2.3.2. COB#2: is the 4-character Country-of-Birth Code of one of the comparands.

4.17.2.3.3. COBVAL: is the decimal value assigned through the HCOB and other COB Category Data Stores that are culture-specific (as they are developed). A default value will be assigned for those COBs that do not enter into special relations. The COBVAL indicates the degree of relationship between the two COBs.

4.17.3. **Purpose**

The COBPROX Data Store provides information on the relative value of the COBs in two comparands. This value can serve to limit the COBs that are accessed for retrieval.

4.17.4. **Function**

The COBPROX is populated by the HCOB and any other partition-specific Country-of-Birth Category Data Stores. The COBPROX provides COB relationship information.

4.18. HISPANIC COUNTRY-OF-BIRTH CATEGORY DATA STORE DECOMPOSTION

4.18.1. **Identification**

This data store is known as the Hispanic Country-of-Birth Category Data Store (HCOB).

4.18.2. **Type**

This HCOB is a data store that will serve as the source of information for the COBPROX Data Store, populating the COBVAL, and will provide the COB Category (COBCAT) necessary for the Hispanic Filter and Sorter.

Figure 80: Design: Hispanic Country-of-Birth Category Data Store (HCOB)

DATA FIELD	DATA TYPE	FIELD SIZE	VALUE RANGE	DATA VALUE
COB#1	characters	4	alphabetics	COB Code
COB#1	characters	4	alphabetics	COB Code
COBCAT	characters	5	alphanumberics	COB1...COB99
COBVAL	decimal	4	0.00...1.00	Various

Figure 81: Example: Piece of HCOB (Values for example only.)

COB#1	COB#2	COBCAT	COBVAL
AGS	AGS	COB1	1.00
AGS	MEX	COB2	0.95
AGS	COL	COB4	0.65
AGS	VENE	COB4	0.65

AGS	PORT	COB5	0.60
COL	COL	COB1	1.00
COL	MEX	COB4	0.65
COL	VENE	COB3	0.85
COL	PORT	COB5	0.60
MEX	MEX	COB1	1.00
MEX	VENE	COB4	0.65
MEX	PORT	COB5	0.60
VENE	VENE	COB1	1.00
VENE	PORT	COB5	0.60
PORT	PORT	COB1	1.00
...			

4.18.3. Definitions

4.18.3.1. COB#1: is the 4-character COB Code of one of the comparands.

4.18.3.2. COB#2: is the 4-character COB Code of one of the comparands.

4.18.3.3. COBCAT: is the category assigned to the relationship of two COBs.

4.18.3.3.1. Categories might be defined as Exact, State, Geographic Region, Dialect Region.

4.18.3.3.2. All relationships are adjustable.

4.18.3.3.3. Example COB Categories are:

- **COB1: Exact** represents an exact match of the COBs: AGS/AGS; the COBPROXVAL would be 1.00.
- **COB2: State Relationship** represents the set of COBs that are states within one country (currently only the Mexican States qualify). The score would be something less than that applied to an exact match but nonetheless high: 0.95.
- **COB3: Northern South America** represents the set of COBs that are in close geographic proximity and share naming conventions: COL/VENE. The value assigned would be less than that for COB2: 0.85.
- **COB4: All Latin America** refers to all COBs in Central and South America and the Spanish-speaking Caribbean. The value assigned would be less than that for COB2: 0.65.
- **COB5: Similar** refers to COBs that have qualified as Hispanic but may not exhibit Hispanic naming conventions: Brazil, Portugal.
- **COB6: All** refers to all COBs and is assigned a value that will allow the search of all COBs; it would be the lowest decimal value used.

4.18.3.4. COBVAL: is the decimal value that will be assigned to a particular COB relationship; this value will be used to determine the COBs that will be permitted in the retrieval process.

4.18.4. **Purpose**

Pre-defined COB category relationships will provide a definition of the values that appear in the COBPROX Data Store.

4.18.5. **Function**

These COB categories will provide information about COB relationships that will contribute to determination of the Composite Score in the Arabic Filter and Sorter.